Machine Learning Models for Spin Excitations

Eoin O'Connor

School of Physics, Trinity College Dublin

College Green, Dublin 2, Ireland

Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

A report submitted to the University of Dublin for a final year project

School of Physics

Academic Advisor: Stefano Sanvito

March 2019

Declaration of Authorship

I, Eoin O'Connor, declare that this thesis titled, Machine learning models for spin excitations and the work presented in it are my own. I confirm that:

- I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at http://www.tcd.ie/calendar.
- I have also completed the Online Tutorial on avoiding plagiarism Ready Steady Write, located at http://tcd-ie.libguides.com/plagiarism/ready-steady-write.

Signed: _____

Date: _____

Abstract

Many-body lattice models generally have Hamiltonians whose associated Hilbert space has a dimension that scales exponentially with the number of lattice sites. For instance, in the Heisenberg model for spin $\frac{1}{2}$ particles, the Hilbert space scales as 2^n where n is the number of lattice sites. By using deep learning neural networks I am able to create an energy functional analogous to the Hohenberg-Kohn energy functional for some specific cases of the Heisenberg model. I demonstrate an alternative method to the second Hohenberg-Kohn theorem for finding the ground-state distribution. Additionally I prove some analytic properties for the generalised Heisenberg model. Finally I investigate using neural networks to solve the generalised Heisenberg model in the absence of an external potential.

Acknowledgements

I would like to thank my advisor, Prof. Stefano Sanvito for his continued help throughout the year. He provided the initial idea for the project and led me in new and interesting directions when I became stuck. I would also like to thank him for proof reading this report.

I would like to thank James Nelson, James is a PhD student of Prof. Sanvito. James gave a lecture and provided notes on using machine learning models. These notes were used as the basis for creating all the machine learning models used in this project. I would also like to thank James for answering any questions I had regarding machine learning models and density fucntional theory.

I would like to thank Eric Prehn for proof reading this report.

Contents

Introduction and Theory

$$\hat{H} = \sum_{ij} J_{ij} \hat{\mathbf{S}}_i \cdot \hat{\mathbf{S}}_j + \sum_i \hat{\mathbf{S}}_i \cdot \mathbf{B}_i \tag{1}$$

The Heisenberg model is a simple model Hamiltonian that describes the interactions of spins in a magnetic system. The model was orignally proposed by Heisenberg in 1926 to explain ferromagnetism in transition metals[1][2]. $\hat{\mathbf{S}}_i$ is the spin angular momentum operator for spin site i. The boldface implies that $\hat{\mathbf{S}}_i$ is a vector containing $\hat{S}_{xi}$, $\hat{S}_{yi}$ and $\hat{S}_{zi}$ operators. In the case where the spin of the site is $\frac{1}{2}$, the spin operators are proportional to the Pauli matrices.

The goal of this project is to find the energy spectrum of this Hamiltonian, in particular the ground state energy. Each term in the Hamiltonian is a tensor product state over all spin sites. We can represent this tensor product state as a matrix using the Kronecker product. This means a matrix of size $m = (2s+1)^n$ is obtained, where n is the number of sites and s is the spin of the sites. Numerical computation to find the energy spectrum (i.e. the eigenvalues of the matrix) scales as m^3, therefore the worst case time complexity of this problem is $(2s+1)^{3n}$. This means it is not computationally feasible to compute the full energy spectrum exactly for any more than a small number of spins.

In order to overcome this scaling challenge we plan to use density functional theory, specifically by adapting the Hohenberg-Kohn (HK) theorems to the Heisenberg Hamiltonian. The HK theorems concern a collection of an arbitrary number of electrons moving under the influence of an external potential $v(r)$. The Hamiltonian of this system takes the form $H = T + V + U$. T is the kinetic energy, V is the potential energy for the nuclei and U is the electron-electron interaction energy. T and U are said to be universal because they are the same for any N-electron system. The first HK theorem states that, assuming a non-degenerate ground-state, the ground-state energy of the system can be expressed as a functional of the electron density $n(r)$. The second HK theorem states that

this ground-state energy can be obtained variationally by minimising the functional over all $n(r)$[3]. This means that the electron density that minimises the functional is the ground state density. One such adaptation of the HK theorems to the Heisenberg model was proven in 2003[2].

While these theorems show the existence of such a functional, they do not provide any information about its form. This project aims to use machine learning (ML) algorithms, specifically deep learning neural networks (NNs) to approximate the functional for the Heisenberg model.

In the case of the HK theorems for the Heisenberg model the spin expectation value $\mathbf{S}_i = \langle \Psi | \hat{\mathbf{S}}_i | \Psi \rangle$ plays the role of the electron density $n(r)$ in the original theorem. The magnetic field $\mathbf{B}_i$ plays the role of the external potential. There are a number of properties of the Heisenberg Hamiltonian that differ from the many electron Hamiltonian and these affect the generality of the theorem for the Heisenberg Hamiltonian. Firstly, the spin interaction term in the Hamiltonian $\sum_{ij} J_{ij} \hat{\mathbf{S}}_i \cdot \hat{\mathbf{S}}_j$ is not universal in the same way that T and U are in the many electron Hamiltonian. Therefore, changing any one of the interaction strengths in the system is going to change the functional. So the functional is not universal except when J_{ij} are kept fixed. Secondly, the HK theorems only apply to non-degenerate states but the Heisenberg model has a large amount of inherent degeneracy. Each energy eigenvalue has $2s + 1$ degeneracy in the absence of a magnetic field. Therefore the existence of such a functional $E[\mathbf{S}_i]$ is only possible when the ground state has spin 0 (i.e. it is non-degenerate).

Since $\sum_{ij} J_{ij} \hat{\mathbf{S}}_i \cdot \hat{\mathbf{S}}_j$ is not universal we must choose specific values of J_{ij} and stick with these. We chose to consider the antiferromagnetic case of a 1 dimensional ring with nearest neighbour interactions. This case is particularly useful for a number of reasons. The ground-state is always spinless when n is even, good approximations are known for the ground-state energy for any number of spins n and the exact solution is known in the thermodynamic limit $n \to \infty$ [4][5]. The first HK theorem then implies that the ground-state energy can be written as $E[\mathbf{S}_i] = F[\mathbf{S}_i] + \sum_i \mathbf{S}_i \cdot \mathbf{B}_i$. The second HK theorem implies that given some external potential $\mathbf{B}_i$ the ground-state spin distribution can be found by minimising $E[\mathbf{S}_i] = F[\mathbf{S}_i] + \sum_i \mathbf{S}_i \cdot \mathbf{B}_i$.

Each functional is only valid for a very specific Hamiltonian and will change whenever the number of spins or their relative interaction strengths are changed.

Therefore, in order to obtain a more general solution a different approach is needed. In the absence of a magnetic field the only variables in the Hamiltonian are the J_{ij} interaction strengths. These can be thought of as a symmetric n x n matrix with a 0 diagonal. Therefore there are $\frac{n(n-1)}{2}$ independent values. It should be possible to create a ML model to predict various properties of the Hamiltonian using J_{ij} as the inputs.

There are three main types of ML. Supervised, unsupervised and reinforcement learning. In supervised learning the model is given a set of input data and the corresponding correct output data. The model attempts to learn how to predict the correct output data for any given input data. In unsupervised learning no output data is given and the model attempts to find patterns in the input data. Finally in reinforcement learning the model attempts to learn the series of actions which maximise reward e.g. learning the best moves to make in a chess game. All the models discussed in this report use supervised learning.

There are two different kinds of supervised learning tasks solved in this project. The main task solved is regression. In regression the model attempts to learn the output of a function $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$. So in this project one such model would be the functional $E[\mathbf{S}_i]$ mapping from $\mathbb{R}^n \rightarrow \mathbb{R}$. The other task considered is classification. Classification is very similar to regression except the output is discrete. The model predicts which of k categories the inputs belong to; $f : \mathbb{R}^n \rightarrow \{1, ..., k\}$. In this project the example of classification is the prediction of the spin of the ground-state energy. There are only $\lfloor \frac{n}{2} \rfloor + 1$ possible spin states for a system with n spins. One possible variation of this is to output a probability distribution rather than a single category[6]. The set of input vectors that are fed into the ML model is called the design matrix. The corresponding matrix of correct output vectors is called the target matrix.

Theoretical and Computational Method

In order for the ML model to learn it needs to be provided with exact numerical solutions. I wrote a python script to calculate the Hamiltonian matrix for given J_{ij} and B_i. The eigenvalues were then calculated using Numpy's linear algebra library. I also wrote a function to calculate the spin expectation value of a given eigenstate.

I extended the code to generate a large number of solutions for random interaction strengths. This data was then used as the input to the ML models. In order to learn about how to make ML models a talk was organised with James Nelson, one of Prof. Sanvito's PhD students. I also read the relevant parts of the book "Deep Learning"[6].

Initially I made a simple regression model using *SKLearn* and a NN model using *Keras* running on a *TensorFlow* backend. I then extended the NN model to solve various problems such as finding the lowest eigenvalue, the full spectrum of eigenvalues, the unique eigenvalues and the spin of the lowest eigenvalue. The Hohenberg-Kohn functional was also calculated using a NN.

In order to evaluate the performance of a ML model it must be tested on a dataset independent from that on which it was trained. The design and target matrices are split up into 3 distinct subsets. The largest subset is the training set, the model uses this set to learn. The smallest subset is the validation set, this set is used to prevent over-fitting of the model. The final set is the test set. This set is used to analyse the overall performance of the model and predict how well it will perform on unseen inputs. In this project I chose to put 64% of the data in the training set, 16% in the validation set and the remaining 20% in the test set.

Results and Discussion

Machine Leaning

There are a number of choices to make when creating a NN. The first is the number of layers in the network. I chose to have only three layers; the input layer, one hidden layer and the output layer. Additional hidden layers can be added but I found that these extra layers had very little effect on the accuracy of the model. The second choice to make is the number of units in each layer. In figure 1 each circle represents one unit and each arrow represents a weight. These weights are adjusted as the model learns. General guidelines suggest that the input layer should have the same number of units as the dimension of your input. In the case of the energy functional the dimension of the inputs is n, the number of spins. In the more general case this is $\frac{n(n-1)}{2}$. The output layer must have the same number of units as the dimension of your target vector and the hidden layer should be greater than the output layer and less than twice the input layer[7]. For the energy functional I found that a NN with n units in the input and hidden layers was sufficient to solve the problem. In the general problem, while the J_{ij} values are the only variables, they don't quite capture the full difficulty of the problem. I found that having $\frac{n(n-1)}{2}$ units in the input and hidden layer was not sufficient to solve the problem. I found that having a number of units similar to the dimensions of the matrix (i.e 2^n) was sufficient in most cases.

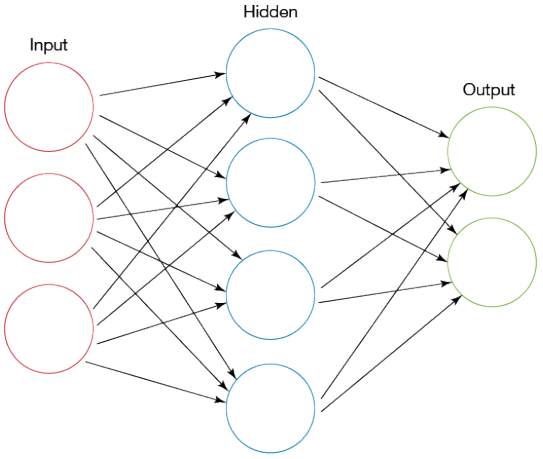

Figure 1: A simple neural network

One also needs to decide how many epochs to train the model for. One epoch is one full loop over the training set. More epochs generally result in a lower error but after a certain point the model is no longer learning and is just overfitting the training data. A balance must be found between the accuracy of the model and the time it takes to train. Another choice to be made is the activation function to use in each layer. I chose to use the rectified linear unit for the input and hidden layers and the linear activation function for the output layer. The rectified linear unit is the function $max(0, x)$, it has been demonstrated to enable better training of NNs than it's counterparts in the majority of problems[8]. The final choices to make are the optimiser and the loss function. Most of the optimisers are variants on stochastic gradient descent. I found that the *adam* optimiser had the best overall performance. The loss function used depends on the type of problem you are solving. For a regression problem the mean-squared error is usually used, for a classification problem cross-entropy is used.

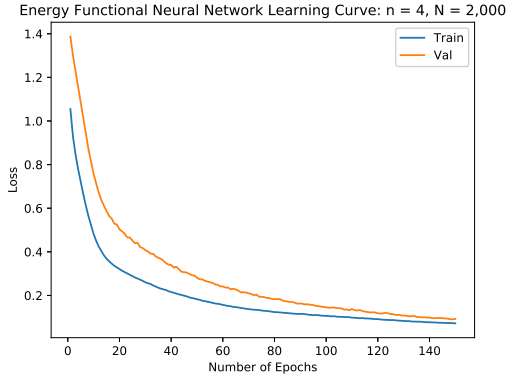

Figure 2: Learning curve of a NN with 2,000 input samples

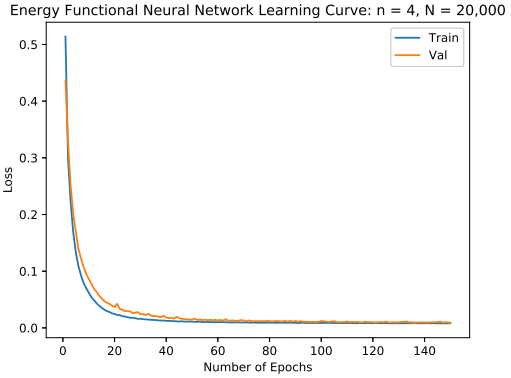

Figure 3: Learning curve of the same NN with 20,000 samples

In the above figures we see how the NN learns over 150 epochs with different training set sizes. The "Train" curve represents the error of the training set and the "Val" curve represents the error of the validation set. In Figure 2 the number of inputs is small and the model takes a much larger number of epochs to learn. Whereas in Figure 3 the model has already converged to a high enough accuracy by around 80 epochs. So while a larger number of samples means that each epoch takes longer it also means that the total number of epochs needed is smaller.

The Hohenberg-Kohn Functional

The Hamiltonian considered in this section is $\hat{H} = \sum_{<ij>} \hat{\mathbf{S}}_i \cdot \hat{\mathbf{S}}_j + \sum_i \hat{S}_{zi} B_{zi}$. Here $\sum_{<ij>}$ indicates that the interaction is nearest neighbour only. The problem is one dimensional and has periodic boundary conditions meaning it is like a ring of n spins. $\hat{S}_{zi} B_{zi}$ indicates that the external potential only acts on the z spin component, in the remainder of this section we will just use B_i to mean B_{zi}. In the Hamiltonian the only variables are the B_i values so $E^{GS} = E[\{B_i\}]$. Therefore it should be possible to create a ML model to predict the ground-state energy for any $\{B_i\}$. For the $n = 3, 4, 5, 6$ and 8 cases $100,000$ samples were generated and solved exactly. These samples were then used as the design and target matrices for multiple ML models so that their abilities could be compared on equal footing. The values of B_i were randomly generated using a uniform distribution between -0.1 and 0.1. The B_i were limited to these values in order to guarantee that the ground-state will never change spin. The model succeeds in learning $E[\{B_i\}]$. We define $\Delta E = E[\{B_i\}] - E[\{0\}]$, then in the $n = 6$ case $\Delta E \in [-0.05, 0]$. In a network with 64 units in the input and hidden layers the mean absolute error (MAE) was $5 \cdot 10^{-5}$. This gives a percentage error of 1.2%. The percentage error is calculated by dividing the MAE by the average of the absolute value of all ΔE. A model with only 6 units in each layer was able to achieve a percentage error of 6%. The percentage errors seem quite high because the average ΔE is very small for even n. $|\Delta E|$ is on average an order of magnitude larger for odd n. This is because the external potential will have a greater effect on a spin $\frac{1}{2}$ state than on a spin 0 state.

Now that we have confirmed that a NN can learn $E[\{B_i\}]$ we want to make a model that takes the ground-state spin distribution $S_i = \langle \Psi_0 | \hat{S}_{zi} | \Psi_0 \rangle$ as the input. This model attempts to learn the functional $F[\{S_i\}] = E[\{S_i\}] - \sum_i S_i \cdot B_i$. The model was successful in learning the functional for even n, this makes sense because when n is odd the ground-state is degenerate so the Hohenberg-Kohn theorem does not apply. In the $n = 6$ case a network with 64 units in the input and hidden layers achieved a MAE of $4 \cdot 10^{-6}$. This gives a percentage error of 0.08%. A model with only 6 units in each layer was able to achieve a percentage error of 0.10%. Clearly the NN is able to learn the functional $E[\{S_i\}]$ and this functional performs better than the functional $E[\{B_i\}]$.

Figures 4 and 5 confirm that the energy functional has been learned to a very high accuracy in both cases. The more complex NN is able to smooth out some

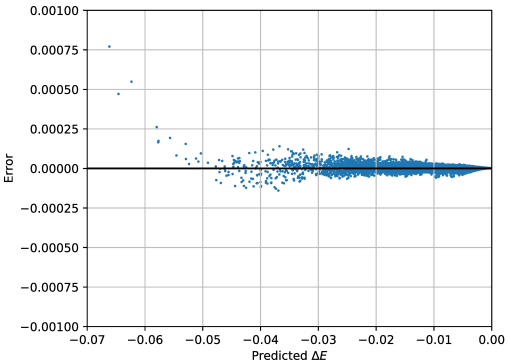

Figure 4: Error in $E[\{S_i\}]$ for n = 8 with 8 units in each layer

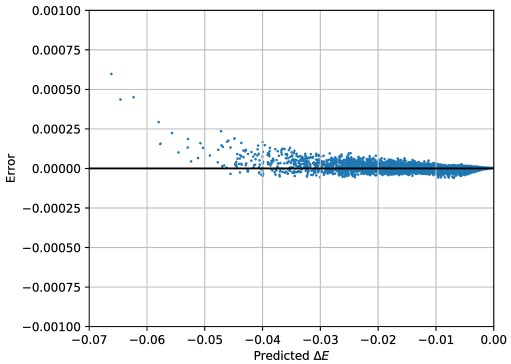

Figure 5: Error in $E[\{S_i\}]$ for n = 8 with 256 units in each layer

of the errors in the smaller network but both still achieve very high accuracy. The errors of all models are shown in Table 1.

n	Input	Units	# Epochs	MAE	% Error
3	B_i	3	15	0.00683	15
3	B_i	8	15	0.0004	0.9
4	B_i	4	15	0.0003	10
4	B_i	16	15	$5 \cdot 10^{-5}$	1.8
4	S_i	4	20	$3.2 \cdot 10^{-6}$	0.10
4	S_i	16	20	$1.5 \cdot 10^{-6}$	0.05
5	B_i	5	15	0.001	2.3
5	B_i	32	15	0.0004	0.9
6	B_i	6	20	0.0003	6
6	B_i	64	20	$6 \cdot 10^{-5}$	1.2
6	S_i	6	20	$5.2 \cdot 10^{-6}$	0.10
6	S_i	64	20	$4 \cdot 10^{-6}$	0.08
8	B_i	8	20	0.0005	7
8	B_i	256	20	$9 \cdot 10^{-5}$	1.2
8	S_i	8	30	$1.3 \cdot 10^{-5}$	0.18
8	S_i	256	30	$7.6 \cdot 10^{-6}$	0.10

Table 1: Errors for all $E[\{B_i\}]$ and $E[\{S_i\}]$

Table 1 further demonstrates that $E[\{S_i\}]$ achieves a greater accuracy than $E[\{B_i\}]$. $E[\{B_i\}]$ does have the significant advantage of working for odd n. The error of $E[\{B_i\}]$ is much higher for odd n but the percentage error is very similar for both odd and even n. This is because the range of possible ΔEs and the average $|\Delta E|$ is much larger for odd n. While the error does in general increase as n gets larger the increase is not too significant. This suggests that it may be possible to generalise the model to arbitrary n values.

The existence of this energy functional is necessary to show that the first Hohenberg-Kohn theorem holds but it is not sufficient in itself to prove that the theorem holds. The proof on the theorem requires that $\{S_i\}$ completely determines $\{B_i\}$. So it should be possible to create a NN that can predict $\{B_i\}$ given $\{S_i\}$. This was found not to be the case. A NN was easily able to solve the reverse problem of finding $\{S_i\}$ given $\{B_i\}$. This suggests that there may be some extra degeneracy appearing even in the spin 0 case.

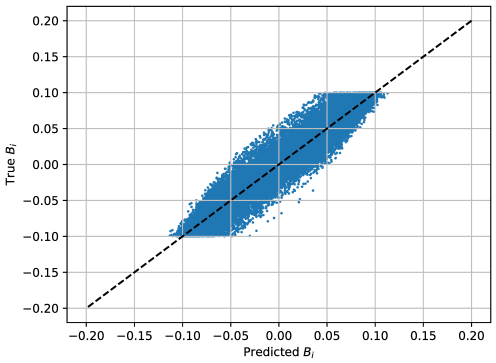

Figure 6: Neural network predicting $B_i[\{S_i\}]$, $n = 8$

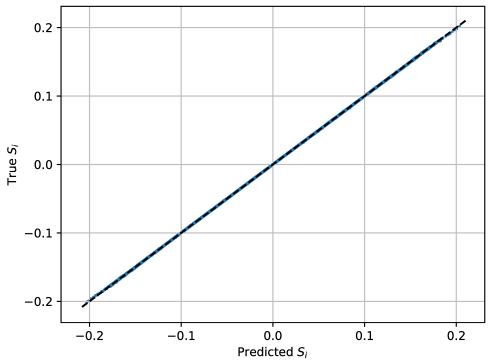

Figure 7: Neural network predicting $S_i[\{B_i\}]$, $n = 8$

The exact same NNs were used in figures 6 and 7 except the design matrix and the target matrix were switched. This demonstrates conclusively that it is not possible in general to find $\{B_i\}$ given $\{S_i\}$. Therefore the it is likely that the Hohenberg-Kohn theorem does not hold even for this special case of a spin zero ground-state.

Now that we have created an energy functional $E[\{S_i\}]$. The next question is how we find $\{S_i\}$? It is the ground-state spin distribution so it cannot be measured experimentally without knowing the ground-state already. All that we know about the system is the external potential $\{B_i\}$. There are two possible ways to obtain the ground-state spin distribution. One is to create a NN to predict $\{S_i\}$ from $\{B_i\}$ and then sub into $E[\{S_i\}]$. This method actually works very well. For the $n = 4$ case with the larger NN the model was able to reduce the error $E[\{B_i\}]$ by ~80% from $5 \cdot 10^{-5}$ to $1 \cdot 10^{-5}$. For n=6 the error remains the same at $6 \cdot 10^{-5}$ and for $n = 8$ the error increases slightly by around 20%. Overall the methods are very similar in performance. Using this two network method has the added benefit of calculating $\{S_i\}$ in the process which is a useful quantity to know.

The other way to obtain the ground-state $\{S_i\}$ is to use the second Hohenberg-Kohn theorem. The theorem implies that given some external potential $\{B_i\}$, the minimum of $E[\{S_i\}] = F[\{S_i\}] + \sum_i S_i B_i$ occurs when $\{S_i\}$ is the ground-state spin distribution. The minimum of the functional can be found using gradient descent. The gradient descent algorithm uses the fact that $E[\{S_i\}]$ decreases fastest in the direction of the negative gradient. So by calculating $\{S_i\}_{n+1} = \{S_i\}_n - \gamma \nabla E[\{S_i\}]$ for small enough $\gamma \in \mathbb{R}_+$ this will converge to the minimum. The derivative of $E[\{S_i\}]$ is given by.

$$\frac{\partial E}{\partial S_i} = B_i + \frac{\partial F[\{S_i\}]}{\partial S_i} \tag{2}$$

$\frac{\partial F[\{S_i\}]}{\partial S_i}$ was then calculated using the central difference method $f'(x) \approx \frac{f(x+h)-f(x-h)}{2h}$. There are some constraints that must hold for S_i at each step in the gradient descent. $\sum_i S_i = 0$, this ensures that the z component of the spin angular momentum for the whole system is zero. Also, all S_i must lie between $-s$ and $+s$. In this case $s = \frac{1}{2}$. In practise this method was unable to find the ground-state spin distribution by minimisation. In fact I was able to show by calculating values of the energy functional close to the ground-state that the ground-state is not the minimum of the energy functional. There are a few reasons why this may be the case. Given that it is unlikely that the first Hohenberg-Kohn theorem holds in this case then there is no reason to assume that the second theorem would hold either. Another explanation is that there are additional constraints on the values of S_i that are not immediately obvious. For example while the above constraint states that all S_i must lie between $-\frac{1}{2}$ and $+\frac{1}{2}$ in the sample of $100,000$ ground-state spin distributions, all S_i were found to lie between -0.13 and $+0.13$.

A More General Solution

Analytic Properties

One property of the Heisenberg Hamiltonian for spin $\frac{1}{2}$ particles with no magnetic field is that there is a large amount of inherent degeneracy. This can be seen clearly in the figure 8 on the next page. The spin of the eigenvalue s is on the x-axis and the number of spin sites in the system n is on the y-axis. The numbers in the circles represent the number of unique eigenvalues with spin s in a system with n spins. Each unique eigenvalue has $2s+1$ degeneracy. If we consider an eigenstate with spin s in a system with n spins then if an additional spin $\frac{1}{2}$ particle is added, this new system has $n+1$ spins and the spin s eigenstate can become either a spin $s-\frac{1}{2}$ or a spin $s+\frac{1}{2}$ eigenstate.Therefore the number of unique eigenvalues is equivalent to the number of unique paths from the first node on the graph to the (s,n) node on the graph. In the case of $s=0$ and general n the numbers of unique eigenvalues are given by the Catalan numbers: $\binom{2m}{m}$-$\binom{2m}{m+1}$. In this case $m=\frac{n}{2}$. All the other numbers in the circles can be written as $\binom{n}{\frac{n}{2}+s}$-$\binom{n}{\frac{n}{2}+s+1}$. Proof of these statements is included in the appendix. Clearly if we add up all the numbers in a row all terms cancel except $\binom{n}{\frac{n}{2}+s_{min}}$ and $\binom{n}{\frac{n}{2}+s_{max}+1}$. $s_{max}=\frac{n}{2}$ therefore the second term is always 0. s_{min} is 0 if n is even and and 0.5 if n is odd. Therefore the number of unique eigenvalues in an n spin system is $\binom{n}{\lceil\frac{n}{2}\rceil}$.

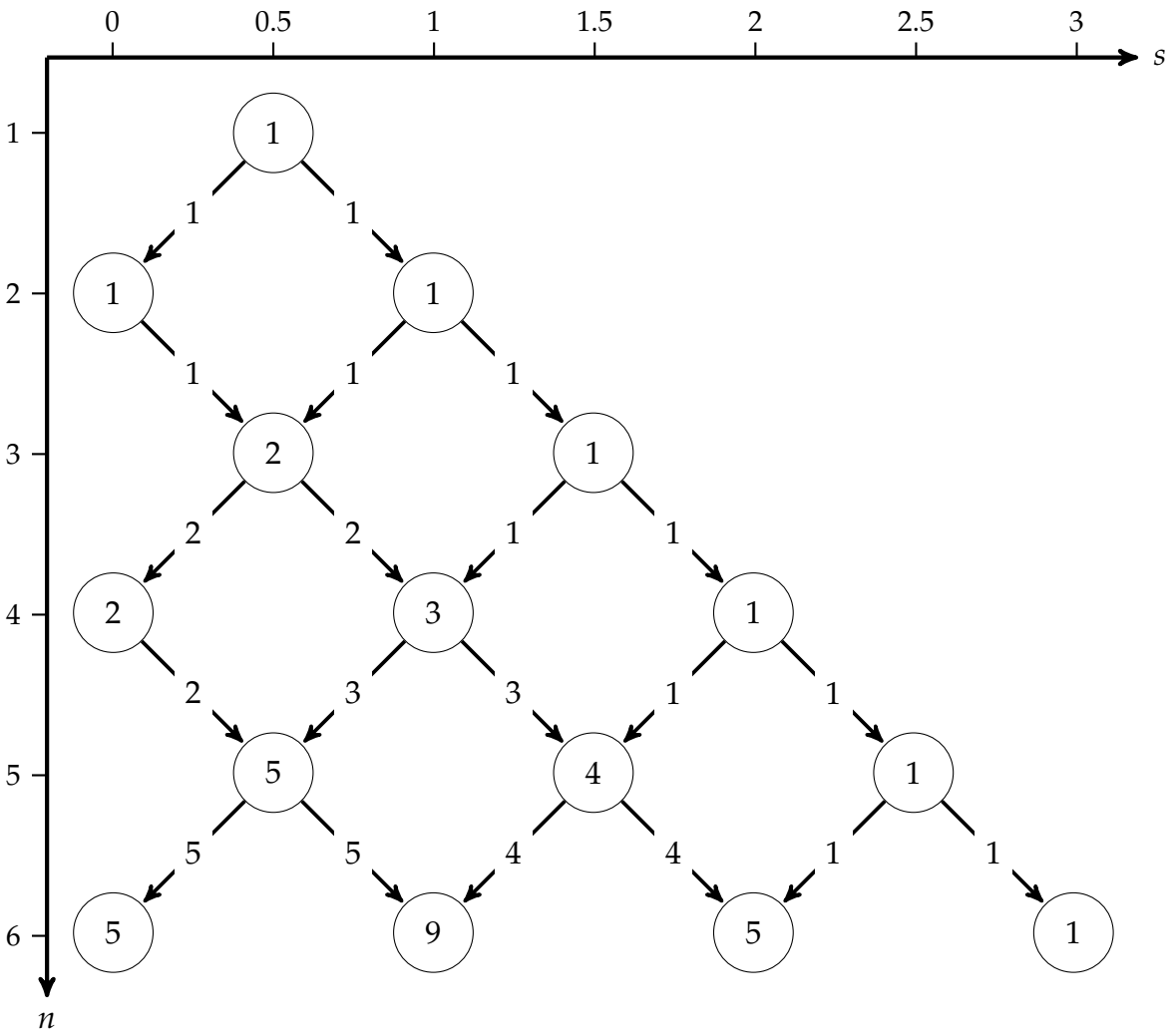

Figure 8: Number of unique eigenvalues in a system with n spin sites for each possible value of spin s. Each unique eigenvalue has a degeneracy of $2s + 1$

In the case of a uniform magnetic field B which acts on the S_z spin component the energy eigenvalues transform quite simply with a change in the magnetic field.

$$\begin{aligned} E_{s,m} &= \langle s, m_z|\hat{H}|s, m_z\rangle = \langle s, m_z|\sum_{ij} J_{ij}\hat{\mathbf{S}}_i \cdot \hat{\mathbf{S}}_j|s, m_z\rangle + \langle s, m_z|\sum_i \hat{\mathbf{S}}_i \cdot \mathbf{B}_i|s, m_z\rangle \\ &= \langle s, m_z|\sum_{ij} J_{ij}\hat{\mathbf{S}}_i \cdot \hat{\mathbf{S}}_j|s, m_z\rangle + \langle s, m_z|B\sum_i \hat{S}_{i,z}|s, m_z\rangle \\ &= \langle s, m_z|\sum_{ij} J_{ij}\hat{\mathbf{S}}_i \cdot \hat{\mathbf{S}}_j|s, m_z\rangle + Bm_z \end{aligned} \tag{3}$$

Therefore knowledge of all the unique eigenvalues in the absence of a magnetic field and their corresponding spin is sufficient information in this case to determine all eigenvalues in the presence of a magnetic field.

While analysing the outputs of the numerical solution I noticed some properties of the eigenvalues.

$$\sum_{i=1}^{2^n} E_i = 0 \tag{4}$$

$$\sum_{i=1}^{2^n} E_i^2 = 2^{n-2}(\frac{3}{4} \sum_{\substack{i,j=1 \\ i \neq j \\ i<j}}^{n} J_{ij}^2 + \sum_{i=1}^{n} B_i^2) \qquad \text{for s} = \frac{1}{2} \tag{5}$$

The first property is equivalent to $Tr(\hat{H}) = 0$

Proof. $\hat{H}$ is real and symmetric so it can be diagonalised. Therefore,

$$\begin{aligned} Tr(\hat{H}) &= Tr(P^{-1}DP) \\ &= Tr(PP^{-1}D) \qquad \text{(cyclic permutation of trace)} \\ &= Tr(D) \\ &= \sum_i E_i = 0 \end{aligned}$$

□

Then using the following property of the trace of the Kronecker product, $Tr(A \otimes B) = Tr(A) \cdot Tr(B)$ [9]. It is trivial to see that $Tr(\hat{H}) = 0$ because the $\hat{S}_i$ matrices are traceless.

Using a similar proof to the first property, it can be shown that $\sum_i E_i^2 = Tr(\hat{H}^2)$ I then proved that $Tr(\hat{H}^2) = 2^{n-2}(\frac{3}{4}\sum_{i,j} J_{ij} + nB^2)$. The proof uses the fact that $(A \otimes B)(C \otimes D) = AC \otimes BD$ [9].

Proof.

$$\hat{H}^2 = \sum_{\substack{i,j \\ i\neq j}} \underbrace{J_{ij}^2 \hat{\mathbf{S}}_i^2 \otimes \hat{\mathbf{S}}_j^2}_{①} + \sum_{\substack{i,j,k,l \\ i\neq k \\ j\neq l}} \underbrace{J_{ij}J_{kl}(\hat{\mathbf{S}}_i\hat{\mathbf{S}}_k) \otimes (\hat{\mathbf{S}}_j\hat{\mathbf{S}}_l)}_{②} + \sum_{\substack{i,j \\ i\neq j}} \underbrace{B_i J_{ij} \hat{\mathbf{S}}_i^2 \otimes \hat{\mathbf{S}}_j}_{③} + \sum_{\substack{i,j \\ i\neq j}} \underbrace{B_j J_{ij} \hat{\mathbf{S}}_i \otimes \hat{\mathbf{S}}_j^2}_{④} +$$
$$+ \sum_{\substack{i,j,k \\ i\neq j\neq k}} \underbrace{B_k J_{ij} \hat{\mathbf{S}}_i \otimes \hat{\mathbf{S}}_j \otimes \hat{\mathbf{S}}_k}_{⑤} + \sum_i \underbrace{B_i^2 \hat{\mathbf{S}}_i^2}_{⑥} + \sum_{\substack{i,j \\ i\neq j}} \underbrace{B_i B_j \hat{\mathbf{S}}_i \otimes \hat{\mathbf{S}}_j}_{⑦}$$

Now consider each of the terms separately.

$$\hat{\mathbf{S}}_i = \frac{1}{2}\boldsymbol{\sigma} \qquad \boldsymbol{\sigma} \text{ are the Pauli matrices}$$
$$\hat{\mathbf{S}}_i^2 = \frac{1}{4}\vec{\mathbb{1}}_2$$

$$\begin{aligned} Tr(①) &= Tr(\sum_{i,j} J_{ij}^2 \hat{\mathbf{S}}_i^2 \otimes \hat{\mathbf{S}}_j^2) \\ &= \frac{1}{16}\sum_{i,j} J_{ij}^2 Tr(\vec{\mathbb{1}}_2 \otimes \vec{\mathbb{1}}_2) \\ &= \frac{3}{16}\sum_{i,j} J_{ij}^2 Tr(\mathbb{1}_2 \otimes \mathbb{1}_2) \end{aligned}$$

In reality $\hat{\mathbf{S}}_i \otimes \hat{\mathbf{S}}_j$ actually represents

$$\mathbb{1}_2 \otimes \mathbb{1}_2 \otimes ... \otimes \hat{\mathbf{S}}_i \otimes ... \otimes \hat{\mathbf{S}}_j \otimes ... \otimes \mathbb{1}_2$$

Therefore

$$\begin{aligned} Tr(①) &= \frac{3}{16}\sum_{i,j} J_{ij}^2 Tr(\underbrace{\mathbb{1}_2 \otimes ... \otimes \mathbb{1}_2}_{\text{n}}) \\ &= 2^n \frac{3}{16}\sum_{\substack{i,j \\ i\neq j \\ i<j}} J_{ij}^2 \end{aligned}$$

In the case of ②, $\sigma_a\sigma_b = \delta_{ab}\mathbb{1}_2 + i\epsilon_{abc}\sigma_c$. Therefore since in ② $i \neq k$ and $j \neq l$
$\implies \hat{\mathbf{S}}_i\hat{\mathbf{S}}_k \propto \sigma$
Therefore $Tr(②) = 0$

$$Tr(\textcircled{3}) = Tr(\textcircled{4}) = Tr(\textcircled{5}) = Tr(\textcircled{7}) = 0$$

This is because all the above terms contain an $\hat{\mathbf{S}}_i$ that isn't squared. This is traceless therefore the whole term is traceless.
The magnetic field was only considered to act on in the z component of the spin.

$$\begin{aligned} Tr(\textcircled{6}) &= \sum_i Tr(B_i^2 \hat{S}_{zi}^2) \\ &= \frac{1}{4} \sum_i B_i^2 Tr(\underbrace{\mathbb{1}_2 \otimes ... \otimes \mathbb{1}_2}_{\text{n}}) \\ &= \frac{2^n}{4} \sum_i B_i^2 \end{aligned}$$

Therefore finally

$$Tr(\hat{H}^2) = 2^{n-2} \left(\frac{3}{4} \sum_{\substack{i,j \\ i \neq j \\ i<j}} J_{ij}^2 + \sum_i B_i^2 \right)$$

□

After proving this formula for the $s = \frac{1}{2}$ case I decided to see if it was possible to derive a formula for general s. Equation (2) still holds for general s because $\hat{\mathbf{S}}_i$ is always traceless. Equation (3) becomes

$$\sum_i E_i^2 = 2^{n-2} \left(3 \left(\frac{s}{3}(s+1)(2s+1) \right)^2 \sum_{\substack{i,j \\ i \neq j \\ i<j}} J_{ij}^2 + 2 \left(\frac{s}{3}(s+1)(2s+1) \right) \sum_i B_i^2 \right) \quad (6)$$

Proof is included in the appendix.

In the case of n=3 spins with no B field there are 3 unique eigenvalues. As can be seen in figure 8 there are two spin $\frac{1}{2}$ eigenvalues with degeneracy 2 and one spin $\frac{3}{2}$ eigenvalue with degeneracy 4. Therefore the first constraint becomes $2E_1 + 2E_2 + 4E_3 = 0$ and the second becomes $2E_1^2 + 2E_2^2 + 4E_3^2 = \frac{3}{2} \sum_{i,j} J_{ij}^2$. Given the above two constraints this would imply that these eigenvalues lie on a 1D manifold in 3D space. The first constraint is the equation of a plane

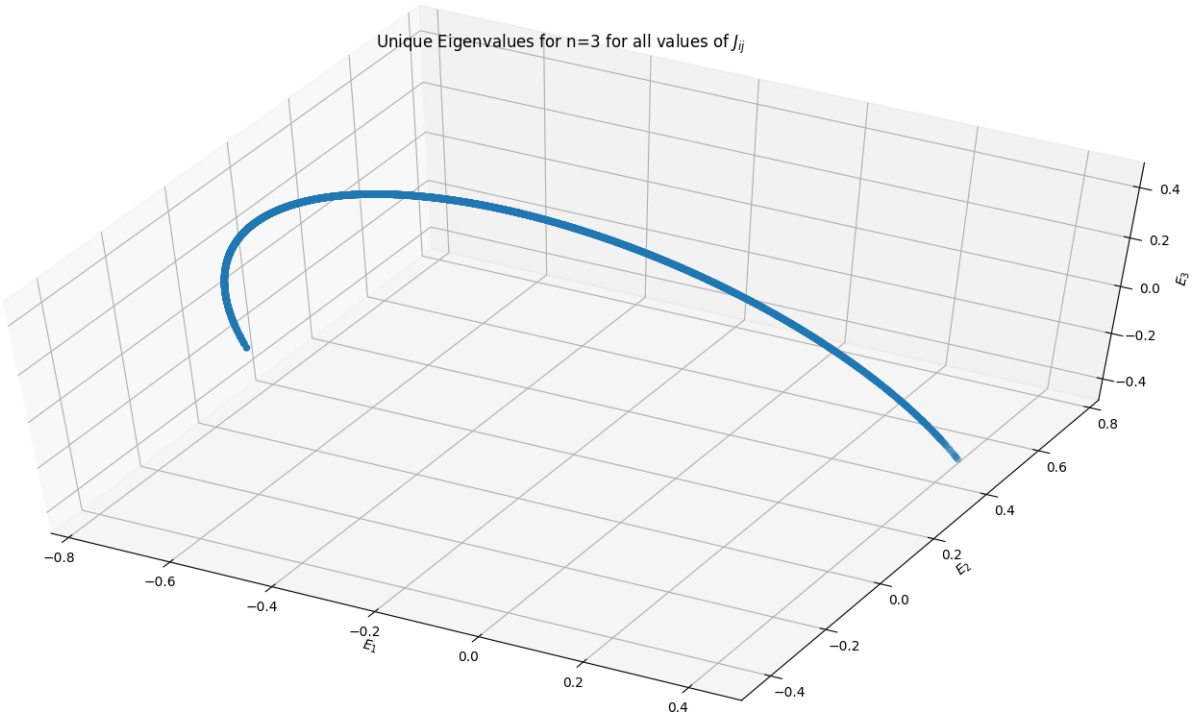

Figure 9: Unique Eigenvalue manifold for $n = 3$

and the second constraint is the equation of an ellipsoid. Therefore we expect the allowed values to lie on an ellipse embedded in 3D. To demonstrate this I plotted the 3 unique eigenvalues for all possible values of J_{ij} with $\sum_{i,j} J_{ij}^2 = 1$. The plot is shown in figure 9. Clearly all the eigenvalues do lie on an ellipse in 3 dimensional space. There must be another constraint that limits the allowed values to half of the ellipse rather than the full ellipse.

Machine Learning the General Case

The simple regression model performed much better than the NN model when the number of spins and the sample size were small. As the number of input vectors and the number of spins in the problem increase the run time of the regression model scales much worse than the NN model. The NN was able to achieve equal or superior accuracy to the regression model in all cases by increasing either the sample size or the number of units in the model. I made the decision to focus on using NNs for the remainder of the project.

I created NNs to solve four main problems.

1. Finding the ground state energy
2. Finding the full energy spectrum
3. Finding only the unique eigenvalues
4. Finding the spin of the ground-state

All of the problems consider the Hamiltonian $\hat{H} = \sum_{ij} J_{ij} \hat{\mathbf{S}}_i \cdot \hat{\mathbf{S}}_j$ where J_{ij} can take any value and the interaction is not limited to nearest neighbours. All particles are spin $\frac{1}{2}$ so $\hat{\mathbf{S}}_i = \frac{\vec{\hat{\sigma}}}{2}$, $\hat{\sigma}$ are the pauli matricies. Due to the similarity of the NNs only two will be discussed in depth, these are the unique eigenvalues which is a regression problem and the ground-state spin which is a classification problem. The others will be included in tables.

One issue that arises in the unique eigenvalues and all eigenvalues problems is that the function that calculates the eigenvalues always returns them in ascending order. One possible alternative is to arrange the eigenvalues in order of increasing spin. There is still some ambiguity in this method because as can be seen in figure 8 there are multiple states with the same spin and different eigenvalues. Rearranging in order of increasing spin for the $n = 4$ unique eigenvalue problem reduced the error by over 50%. It also has the added benefit that now the spins of all of the eigenvalues are known, allowing the effect of applying a uniform magnetic field B to be easily calculated using equation 3. Unfortunately this rearranging does not always improve the accuracy the NN, for example when $n = 6$ the NN fails to learn at all when the eigenvalues are rearranged but can learn without rearranging. I haven't been able to come up with a good reason for why this occurs.

In the unique eigenvalues problem rounding errors become a big issue as n gets larger. This is because rounding errors can sometimes cause eigenvalues that are equal to differ very slightly when solved for numerically. To counteract this we can round the eigenvalues to a number of decimal places before checking if they are equal. At a certain point this rounding can also cause eigenvalues that are not equal to appear equal after rounding. These two rounding issues begin to occur at the same point at around $n = 7$, so after this

higher precision floating point numbers must be used.

Datasets of random J_{ij} of size 50,000 were created for $n = 3, 4, 5$ and 6 and the exact numerical solutions were calculated. These datasets were then used to train a number of NNs in order to compare their accuracy.

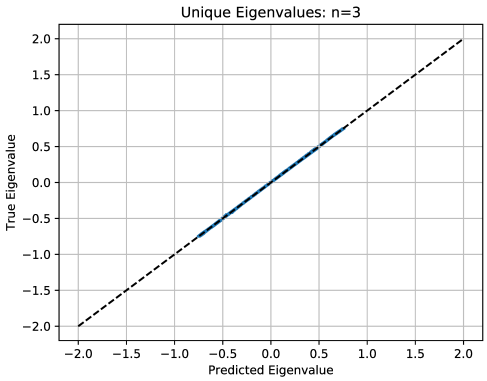

Figure 10: The neural network is able to predict the unique eigenvalues to a very high accuracy when $n = 3$

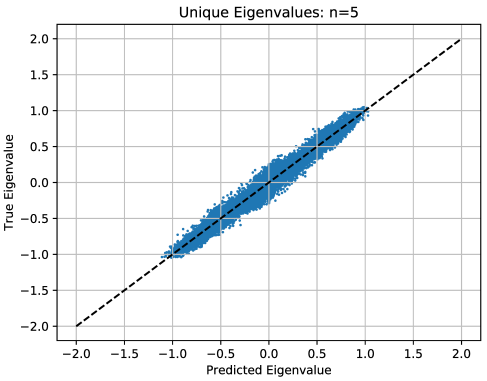

Figure 11: The neural is able to make good estimations for the unique eigenvalues but the errors are noticable for $n = 5$

The NN succeeds in learning the unique eigenvalue problem when n is small as can be seen in figure 10. As n gets larger the model finds it much harder to learn as can be seen in figure 11. Some of this error can be removed by increasing the number of units in each layer of the network. Another way to reduce error is to increase the sample size, the effects of both methods are demonstrated in figures 12 and 13.

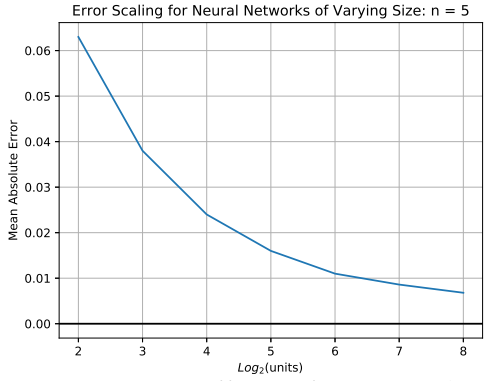

Figure 12: Effects of Varying the number of units in each layer. Sample size N = 50,000, $n = 5$

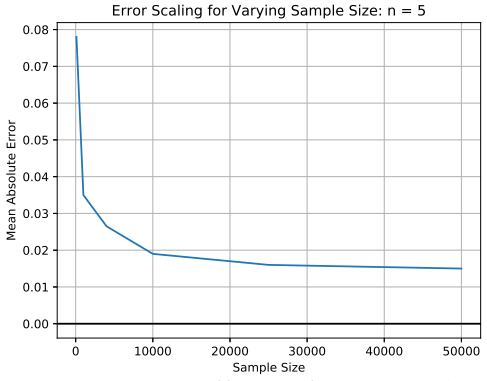

Figure 13: Effects of Varying the input sample size. 2^5 units in each layer, $n = 5$

It is clear from figure 13 that increasing the sample size increases the accuracy of the model but only to a certain point. The MAE appears to asymptote to

greater than 0.01. Figure 12 shows that it is possible to achieve an error less than 0.01 by increasing the number of units in each layer of the NN. Both a sufficiently large sample size and a sufficiently complex NN are needed to solve this problem.

Since multiplying a matrix by a constant has the effect of multiplying the eigenvalues by a constant this means we can rescale all the J_{ij} values by a scalar and this will rescale the eigenvalues by the same scalar. The J_{ij} values are generated using a random normal distribution so $\sum_{ij} J_{ij}^2$ is generally different for any randomly generated set. We can consider only the case in which $\sum_{ij} J_{ij}^2 = 1$ without loss of generality (just multiply these eigenvalues by actual $\sum_{ij} J_{ij}^2$ to obtain correct eigenvalues). This constraint then implies that the number of independent inputs is now one less. I had the idea to use spherical coordinates instead to represent the J_{ij} components because the constraint implies they lie on a hypersphere of radius 1. It is also possible that these spherical coordinates may help to emphasise some of the underlying symmetry seen in equation 5.

n	Output	MAE
3	Unique	0.0030
4	Unique	0.011
5	Unique	0.015
6	Unique	0.042
3	GS	0.0026
4	GS	0.011
5	GS	0.012
6	GS	0.031
3	Full	0.0034
4	Full	0.019
5	Full	0.017
6	Full	0.042

Table 2: Errors for various models with normally distributed J_{ij} input

n	Output	MAE
3	Unique	0.009
4	Unique	0.035
5	Unique	0.031
6	Unique	N/A
3	GS	0.008
4	GS	0.026
5	GS	0.022
6	GS	N/A
3	Full	0.012
4	Full	0.036
5	Full	0.029
6	Full	N/A

Table 3: Errors for various models with uniformly distributed spherical J_{ij} input

Table 2 shows the error in all models in which J_{ij} were the inputs and their values were distributed randomly in a normal distribution. I was unable to generate data for $n = 6$ with spherical inputs because the rounding error challenges mentioned earlier are emphasised when converting from spherical to Cartesian coordinates. In table 3 the inputs to the models were Spherical

coordinates calculated from the same J_{ij} used in Table 2. These tables demonstrate that despite being able to decrease the dimensions of the input vector the spherical coordinates actually have a negative effect on the accuracy of the model. One reason why this might be the case is that the J_{ij} values are generated from a normal distribution with mean 0 and standard deviation 1. NNs perform much better when their inputs have 0 mean and a known standard deviation. This is already true for J_{ij} so this makes it a good candidate for an input vector.

As mentioned before equations 4 and 5 are the equations of a plane and of a sphere respectively. Consider the eigenvalues as a vector $\mathbf{E} = (E_1, E_2, ...)$ of length 2^n. Equation 4 implies that we can make an orthogonal transformation of these coordinates into a plane in which one of the components will always be zero. Since the transformation is orthogonal this means that the metric is conserved and therefore equation 5 still holds in this new coordinate system. These new coordinates can then be converted into spherical coordinates in which the radius is constant and known, reducing the dimensions on the output to $2^n - 2$.

In order to perform the orthogonal transformation you need the corresponding orthonormal basis. The normal vector to the plane is just the coefficients of the equation. In this case $(1, 1, ...)$. I chose to let E_1 transform to become the normal vector. To find a basis of the plane we set $E_i = 1$ and all other $E_j = 0; i \neq j$ and therefore $E_1 = -E_i = -1$. Unfortunately this basis of the plane is not orthogonal or normalised. The Gram-Schmidt process is applied to the basis to make it orthonormal. This new basis is then transformed into $2^n - 1$ dimensional spherical coordinates. For the unique eigenvalue case these transformations are still possible but they are a little more tricky. Equation (2) becomes $\sum_{i=1}^{m} = c_i E_i = 0$. m = $\binom{n}{\lceil \frac{n}{2} \rceil}$ and $c_i = 2s_i + 1$. It makes more sense to rescale this equation to be $\sum_{i=1}^{m} = d_i \lambda_i = 0$ where $d_i = \sqrt{c_i}$ and $\lambda_i = d_i E_i$. Equation (3) then becomes $\sum_{i=1}^{m} \lambda_i^2 = 2^{n-2}(\frac{3}{4} \sum_{i,j=1}^{n} J_{ij}^2 + \sum_{i=1}^{n} B_i^2)$. The normal vector to the plane is then $\mathbf{d}$ and the basis vectors of the plane have the form $\lambda_i = 1$, $\lambda_j = 0; i \neq j$ and $\lambda_1 = -\frac{d_i}{d_1}$. The rest of the procedure follows as before. While this method does allow the length of the target vector to be reduced by 2 it does not improve the error of the model. In fact the error of this model was found to be significantly higher than all other models. The change of basis is a linear transformation and should not cause the error to of

the model to increase. Therefore transforming the eigenvalues into spherical coordinates must be causing the additional error. It is also probable that the values of d_i change for different J_{ij} (i.e. an eigenvalue E_i changes spin) this changing means that the transformation is not consistent and this could cause problems for the neural network.

The final problem I considered was finding the spin of the ground-state. This is fundamentally different from the other models as it is a classification problem. For a system with n spins there are only $\lfloor \frac{n}{2} \rfloor + 1$ possible spins so the space of outputs is not continuous. The loss function used in this model is called *categorical cross-entropy*. Using this loss function requires the target vector to be transformed into a one-hot encoding. In a problem with k categories the one-hot encoding creates a target vector of length k with 0 in all entries except for the entry corresponding to the category which has value 1. The ML model then predicts a vector of probabilities. The softmax activation function is used in the last layer so that the probabilities will all sum to 1. Because the space of outputs is not continuous measures such as mean-squared error don't make sense. Instead the performance of the model is measured by it's accuracy i.e. the percentage of the time it predicts the correct spin. The accuracy of the model for $n = 4$ and $n = 6$ is shown in figures 14 and 15.

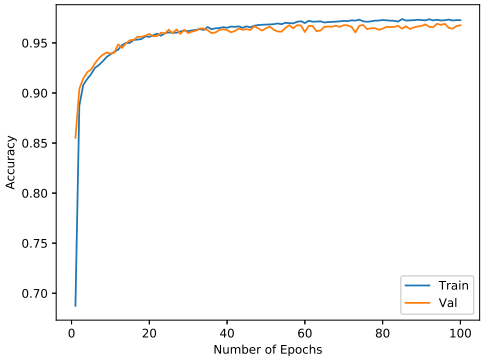

Figure 14: n = 4; 16 units in each layer; 30,000 samples

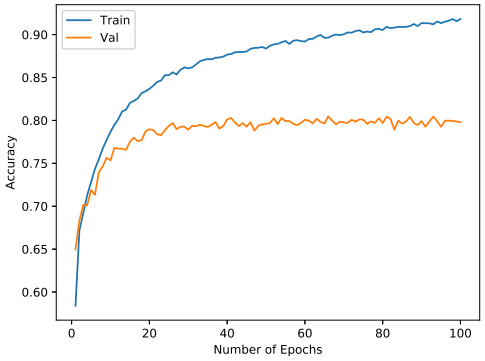

Figure 15: n = 6; 64 units in each layer; 30,000 samples

In the $n = 4$ case the model achieves approximately 97% accuracy and the validation error remains close the the training error. The $n = 6$ model is only able to achieve 78% accuracy and the validation error flattens out very quickly, while the training error continues to decrease. This could suggest that more samples are needed or that a more complex NN may be useful.

Conclusions

While I was unable to confirm that the Hohenberg-Kohn theorems hold for the Heisenberg model, the creation of an energy functional $E[\{S_i\}]$ is still very promising. I was able to demonstrate that it is possible to avoid the second Hohenberg-Kohn theorem and obtain the ground-state spin distribution directly from the external potential. The clear way forward is to attempt to create a general functional for arbitrary n. This is not a straightforward task for neural networks to solve because increasing n increases the dimension of the input vector. This neural network for general n would have to be able to deal with different input vector lengths. One possible way to overcome this problem is to use a recurrent neural network. These kinds of networks can deal with varying input sizes and are currently used in problems such as handwriting recognition[10]. If I had more time I would have like to consider systems with higher spins or systems with different spins at different sites. It would be fairly straightforward to modify the neural networks used for spin $\frac{1}{2}$ to work for higher spin systems.

The general Heisenberg model appears to be too difficult to solve for general n using a neural network. More progress might be found by considering more symmetric cases, such as lattice models with only nearest neighbour or next-nearest neighbour interactions. In such lattice models convolutional neural networks can be applied to help reduce the dimensions of the problem. Such models have already been analysed for the Hubbard model[11]. The analytic properties of the Heisenberg Hamiltonian that were found are very useful. It may be possible to apply these properties to help with the learning of $E[\{S_i\}]$. If I had more time I would have liked to analyse why additional eigenvalue degeneracies are seen in more symmetric problems, such as the spin ring and 2D square lattice models with nearest neighbour interactions.

Bibliography

[1] W. Heisenberg. *Z. Phys*, 38(441), 1926.

[2] Valter L. Lbero and K. Capelle. Spin-distribution functionals and correlation energy of the Heisenberg model. *Physical Review B*, 68(024423), 2003.

[3] P. Hohenberg and W. Kohn. Inhomogeneous electron gas. *Physical Review*, 136(B864), 1964.

[4] Valter L. Lbero and K. Capelle. Density-functional treatment of model hamiltonians. *https://arxiv.org/pdf/cond-mat/0506206.pdf*, 2005.

[5] H. Bethe. On the theory of metals. *Z. Phys*, 71(201), 1931.

[6] Y. Bengio I. Goodfellow and A. Courville. *Deep learning*. MIT Press, Cambridge, 2016.

[7] Jeff Heaton. *Introduction to Neural Networks for Java*. Heaton Research, Inc, 2008.

[8] Antoine Bordes Xavier Glorot and Yoshua Bengio. Deep sparse rectifier neural networks. *Journal of Machine Learning Research*, 15(315), 2010.

[9] R.A. Horn and C.R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, Cambridge, 1991.

[10] S. Fernandez R. Bertolami H. Bunke A. Graves, M. Liwicki. A novel connectionist system for improved unconstrained handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(855), 2009.

[11] Rajarshi Tiwari James Nelson and Stefano Sanvito. Machine learning density functional theory for the hubbard model. *Phys. Rev. B*, 99(075132), 2019.

[12] D. André. Solution directe du probléme résolu par M. Bertrand. *Comptes Rendus de l'Académie des Sciences, Paris*, 105, 1887.

Appendices

Catalan Numbers

Below is the proof that the number of paths that end on the far left of figure 8 is given by the Catalan numbers. This proof uses André's reflection method[12]. This method was originally used to solve Bertrand's ballot problem.

Proof. We can consider an equivalent problem starting at $n = 0$, $s = 0$ since the first step in every path must be to go to $n = 1$,$s = 0.5$ so no extra paths are added. Each step in the path is either to the left or the right. In order to end up at $s = 0$ at the end there must be the same number of steps to the left as steps to the right. Therefore n must be even and the total number of combinations of steps that have equal numbers of left and right is $\binom{n}{\frac{n}{2}}$. Now we need to figure out how many of these paths are invalid i.e they go to negative s. Every invalid path must have $s = -\frac{1}{2}$ at some point, this path up to this point has one more left step than right step. Therefore the remainder of the path has one more right step than left. If we reverse every step in the remainder of the path then it is clear that this path will end at $s = -1$. The number of such paths is then $\binom{n}{\frac{n}{2}+1}$. Therefore the total number of unique eigenvalues with $s = 0$ for given n is $\binom{n}{\frac{n}{2}} - \binom{n}{\frac{n}{2}+1}$

□

The proof that the number of paths to general (n, s) is given by $\binom{n}{\frac{n}{2}+s} - \binom{n}{\frac{n}{2}+s+1}$ is very similar to the above proof and is shown below:

Proof. In this case every path must have $2s$ times more right steps than left steps. Therefore the total number of paths the end at s is $\binom{n}{\frac{n}{2}+s}$. By the same reasoning as the above proof the number of invalid paths is given by $\binom{n}{\frac{n}{2}+s+1}$. Giving the total number of valid paths to be $\binom{n}{\frac{n}{2}+s} - \binom{n}{\frac{n}{2}+s+1}$

□

Proof of Equation 6

Proof of Equation 6 is given below

Proof.

$$\hat{H}^2 = \sum_{\substack{i,j \\ i\neq j}} \underbrace{J_{ij}^2 \hat{\mathbf{S}}_i^2 \otimes \hat{\mathbf{S}}_j^2}_{①} + \sum_{\substack{i,j,k,l \\ i\neq k \\ j\neq l}} \underbrace{J_{ij}J_{kl}(\hat{\mathbf{S}}_i\hat{\mathbf{S}}_k) \otimes (\hat{\mathbf{S}}_j\hat{\mathbf{S}}_l)}_{②} + \sum_{\substack{i,j \\ i\neq j}} \underbrace{B_i J_{ij} \hat{\mathbf{S}}_i^2 \otimes \hat{\mathbf{S}}_j}_{③} + \sum_{\substack{i,j \\ i\neq j}} \underbrace{B_j J_{ij} \hat{\mathbf{S}}_i \otimes \hat{\mathbf{S}}_j^2}_{④} +$$
$$+ \sum_{\substack{i,j,k \\ i\neq j\neq k}} \underbrace{B_k J_{ij} \hat{\mathbf{S}}_i \otimes \hat{\mathbf{S}}_j \otimes \hat{\mathbf{S}}_k}_{⑤} + \sum_i \underbrace{B_i^2 \hat{\mathbf{S}}_i^2}_{⑥} + \sum_{\substack{i,j \\ i\neq j}} \underbrace{B_i B_j \hat{\mathbf{S}}_i \otimes \hat{\mathbf{S}}_j}_{⑦}$$

$$Tr(③) = Tr(④) = Tr(⑤) = Tr(⑦) = 0$$

This is because all the terms contain a $\hat{\mathbf{S}}_i$ that isn't squared. This is traceless therefore the whole term is traceless. $Tr(②) = 0$ by the commutation relations

of the spin operators and similar reasoning to the proof of equation 5.

$$\begin{aligned} Tr(\textcircled{1}) &= Tr(\sum_{i,j} J_{ij}^2 \hat{\mathbf{S}}_i^2 \otimes \hat{\mathbf{S}}_j^2) \\ &= \sum_{i,j} J_{ij}^2 2^{n-2} (Tr(\hat{S}_x^2 \otimes \hat{S}_x^2) + Tr(\hat{S}_y^2 \otimes \hat{S}_y^2) + Tr(\hat{S}_z^2 \otimes \hat{S}_z^2)) \\ &= \sum_{i,j} J_{ij}^2 2^{n-2} (Tr(\hat{S}_x^2)^2 + Tr(\hat{S}_y^2)^2 + Tr(\hat{S}_z^2)^2) \end{aligned}$$

Now consider $Tr(\hat{S}_x^2)$. $\hat{S}_x = \frac{1}{2}(\hat{S}_+ + \hat{S}_-)$ so $\hat{S}_x^2 = \frac{1}{4}(\hat{S}_+^2 + \hat{S}_-^2 + \hat{S}_+\hat{S}_- + \hat{S}_-\hat{S}_+)$
$[\hat{S}_+, \hat{S}_-] = 2\hat{S}_z$ so $\hat{S}_x^2 = \frac{1}{4}(\underbrace{\hat{S}_+^2 + \hat{S}_-^2 + 2\hat{S}_z}_{\text{Traceless}} + 2\hat{S}_-\hat{S}_+)$. Therefore

$$\begin{aligned} Tr(\hat{S}_x^2) &= \frac{1}{2} Tr(\hat{S}_-\hat{S}_+) \\ &= \frac{1}{2} \sum_{m=-s}^{s} < m|\hat{S}_-\hat{S}_+|m > \\ &= \frac{1}{2} \sum_{m=-s}^{s} (s(s+1) - m(m+1)) \\ &= \frac{1}{2}(2s+1)s(s+1) - \frac{1}{2}\sum_{m=-s}^{s} m^2 - \frac{1}{2}\sum_{m=-s}^{s} m \\ &= \frac{1}{2}(2s+1)s(s+1) - \sum_{m=0}^{s} m^2 \\ &= \frac{1}{2}(2s+1)s(s+1) - \frac{s(s+1)(2s+1)}{6} \\ &= \frac{s(s+1)(2s+1)}{3} \end{aligned}$$

Now $Tr(\hat{S}_y^2)$. $\hat{S}_x = \frac{1}{2i}(\hat{S}_+ - \hat{S}_-)$ so $\hat{S}_y^2 = \frac{1}{4}(-\hat{S}_+^2 + -\hat{S}_-^2 + \hat{S}_+\hat{S}_- + \hat{S}_-\hat{S}_+)$
$\implies Tr(\hat{S}_y^2) = Tr(\hat{S}_x^2)$

$$\begin{aligned} Tr(\hat{S}_z^2) &= \sum_{m=-s}^{s} < m|\hat{S}_z^2|m > \\ &= 2\sum_{m=0}^{s} m^2 \\ &= \frac{s(s+1)(2s+1)}{3} \end{aligned}$$

Finally this gives us

$$\begin{aligned} Tr(\textcircled{1}) &= \sum_{i,j} J_{ij}^2 2^{n-2} (Tr(\hat{S}_x^2)^2 + Tr(\hat{S}_y^2)^2 + Tr(\hat{S}_z^2)^2) \\ &= \sum_{i,j} J_{ij}^2 2^{n-2} (3 \left(\frac{s(s+1)(2s+1)}{3} \right)^2) \end{aligned}$$

$$\begin{aligned} Tr(\textcircled{6}) &= \sum_i Tr(B_i^2 \hat{S}_{zi}^2) \\ &= 2^{n-1} \sum_i B_i Tr(\hat{S}_z^2) \\ &= 2^{n-1} \sum_i B_i \frac{s(s+1)(2s+1)}{3} \end{aligned}$$

Combining these two terms gives the final expression

$$\sum_i E_i^2 = 2^{n-2} \left(3 \left(\frac{s}{3}(s+1)(2s+1) \right)^2 \sum_{\substack{i,j \\ i \neq j \\ i<j}} J_{ij}^2 + 2 \left(\frac{s}{3}(s+1)(2s+1) \right) \sum_i B_i^2 \right)$$

□