Ising Model Notes

Lachlan Kan

March 2025

1 Introduction

The Ising model describes the interaction between neighbouring spins on a lattice under an external magnetic field. It is used to study the polarisation of ferromagnetic materials and how they align with an external magnetic field. Let us begin with a 1D chain of lattice sites from site 0 to site N. We know that each site can only interact with its nearest neighbours, and that the energy depends on how different their spins are. To model this, we say that

$$\hat{H}_{int,\ n} \propto \hat{S}_z^n \hat{S}_z^{n+1} \tag{1}$$

Where the product is our measure of their difference in spins. In a lattice with N sites, we can add this up to find the total interaction energy. Notice that we are summing until $N-1$ since the $N-1$ term will automatically interact with the N term, due to the $n+1$ in the operation.

$$\hat{H}_{int} = -J \sum_{n=0}^{N-1} \hat{S}_z^n \hat{S}_z^{n+1} \tag{2}$$

Where the constant of proportionality is the coupling constant J. It is customary to append a negative sign due to convention. Suppose we subject the lattice to an external magnetic field of strength h, which tends to magnetise the lattice, influencing their spins along the x direction. This energy can thus be found by

$$\hat{H}_{ext} = -h \sum_{n=0}^{N} \hat{S}_x^n \tag{3}$$

Where the negative sign is once again due to convention. To find the total energy, we simply sum the energies from (2) and (3), giving

$$\hat{H} = -J \sum_{n=0}^{N-1} \hat{S}_z^n \hat{S}_z^{n+1} - h \sum_{n=0}^{N} \hat{S}_x^n \tag{4}$$

Notice that the hamiltonian consists of two competing terms trying to influence the spins of the qubits: the interaction term and the field term. When $J >> h$, the interaction term takes over, and the spin will be aligned along the z-axis. Conversely, when $h >> J$, the field will be strong enough to overpower the interactions, leaving the spins aligned along the field in the x-axis.

2 Spins Operators of Qubits

Now, to use this hamiltonian, we first have to find each S_z^n associated with the nth site. To do this, we make use of the Pauli z matrix, S_z. For example, for 3 lattice sites, we say that

$$\begin{aligned}\hat{S}_z^0 &= \hat{S}_z \otimes I \otimes I \\ \hat{S}_z^1 &= I \otimes \hat{S}_z \otimes I \\ \hat{S}_z^2 &= I \otimes I \otimes \hat{S}_z\end{aligned}$$

For the case of 3 lattice sites, we have 3 tensor products to obtain each spin. Notice that the spin is only described by the Pauli matrix when its at its own index (The one for spin 0 has $\hat{S}_z$ at the 0th place, the one for spin 1 has $\hat{S}_z$ at the 1st place, etc.). This is equivalent to saying "there is only a component of spin at the site where the index matches the location of the particle". Therefore, the spin operator acts non-trivially only on its corresponding site, and as identity elsewhere. This can be generalised to higher numbers of sites, but generally, for N sites, there will be N tensor products per site to compute the spins. To compute the x spin in the second term, we repeat the above process, replacing the Pauli z matrix with the Pauli x matrix.

3 Energy Levels

To find the energy levels, we need to solve the Schrodinger equation for its energy eigenvalues E, which is as follows

$$\hat{H}\left|\psi\right\rangle = E\left|\psi\right\rangle \tag{5}$$

This is an eigenvector-eigenvalue equation, where $|\psi\rangle$ represents the state of the whole lattice, and $\hat{H}$ is given in (4). Computing the hamiltonian for N lattice sites should give an $2^N \times 2^N$ matrix. This means computing the various energy eigenvalues E will be an exponentially more arduous task with every increasing N. Solving for the eigenvalues E will give the energy levels, while solving for the eigenvectors $|\psi\rangle$ gives the state of the whole lattice.

4 Computing the Spins

The spins of each site can either be up or down. Since the qubits can have spin $S_z^n = \pm 1/2$, spin up will be assigned to $+1/2$ and spin down to $-1/2$. To obtain this spin, we first find the expectation value of the $|\psi\rangle$ computed in the last section when applied to by $\hat{S}_z^n$ (computed in section 2), which gives

$$S_z^n = \left\langle\psi\right| \hat{S}_z^n \left|\psi\right\rangle \tag{6}$$

This gives the z spin for the nth qubit. Applying this formula to each qubit from $n = 0$ to $n = N$ will give the spins to all the qubits. To find the x-spin, the same process can be repeated but with the $\hat{S}_x^n$ computed in section 2.

5 Ising Model in 2 Dimensions

The Ising model can be further generalised into 2 dimensions. Now, for any given site i, the qubit can interact with any of its nearest neighbour sites j. The Hamiltonian now reads

$$\hat{H} = -J \sum_{\langle i,j \rangle} \hat{S}_z^i \hat{S}_z^j - h \sum_i \hat{S}_x^i \tag{7}$$

Each site is specified by a set of coordinates (x, y) since the lattice is now a 2D grid of sites (Note that in the Python code, x and y are replaced by i and j because I've gotten used to using i and j for indices inside loops and I don't intend to change that. The index number in the code is specified by n and the coordinates by i and j. There is no x and y anywhere in the code. Don't go looking for it).

6 The Monte Carlo Method

Obviously, solving (7) via direct diagonalisation is impossible. The 1D strip has a Hamiltonian matrix of size $2^N \times 2^N$, and in 2D, we need to compute not just N, but N^2 sites. Hence the Hamiltonian matrix of (7) will have dimensions $2^{N^2} \times 2^{N^2}$ - a ridiculously large number that even the best supercomputers of the world will run into problems trying to solve for a relatively small lattice. With this, its safe to say that we need a new method of calculating the stable energies and spins of the system. Evidently we can't directly treat all the qubits, but what we can do, is randomly sample *some* qubits and calculating the energy for their interactions. It is easy to see that for a large enough sample size, the interactions of this sampled group will be somewhat representative of the exact solution. This is where Monte Carlo comes in.

6.1 Side Quest - Approximating π

To get a feel for the Monte Carlo method, lets consider a totally separate problem - approximating π. If we draw a square with side lengths r, and then inscribe a quarter circle inside that square, we know that the ratio of areas between the square and quarter circle will be as follows.

$$R = \frac{A_O}{A_\square} = \frac{\frac{1}{4}\pi r^2}{r^2} = \frac{1}{4}\pi \implies \pi = 4R \tag{8}$$

Where A_O denotes the area of the quarter circle, not a full circle. Now if we have the areas of the quarter circle and the square, then the exact value of π can be easily found. However, we don't. So how can we find these areas? One way to approximate the area is to draw the square circle combination on a piece of paper and sprinkle sand evenly onto the paper. When the sand covers roughly the whole sheet, we can then count the grains of sand inside the quarter circle, then count the total grains of sand inside the square, and then divide the two to find the ratio. The reason for doing this is because the number of sand grains the quarter circle can hold is directly proportional to the area of the quarter circle, and the grains of the square is proportional to its area as well, so dividing the two numbers is a good representative of the ratio of their areas. Of course, counting grains of sand is unrealistic (probably a task for a professor's team of graduate students), and so we leave this mundane task to the computer.

To implement this on a computer, we can generate a random set of points $P_i(x_i, y_i)$, which represents the random positions of all the grains of sands. Then, we can compute each points' radial distances from the origin (which we set to be one corner of the paper, at the center of the quarter circle), $d_i = \sqrt{x_i^2 + y_i^2}$. It is easy to see that for the ith point, if it lies inside the circle, then $d_i \leq r$. Conversely, if $d_i > r$, then the point lies outside of the circle. We can thus approximate π by quadrupling the ratio of the number of points of which $d_i \leq r$ to the total number of points.

6.2 The Metropolis Algorithm

To apply this method to our problem, we can randomly sample each lattice site and calculate their interactions. However, we immediately run into a problem - to calculate the interactions of any particular lattice site, we must know the spins of its neighbours! And to obtain the spins of the neighbours, we must calculate their interactions with the neighbours neighbours! And this goes on and on until we have to exactly calculate for the whole lattice - which we cannot do. So how to we calculate the interaction of a single site without knowing the details of its neighbours? This is where the Metropolis Algorithm comes in.

Suppose we have a lattice with sites that has totally random spins. Since we know the spins (they are random), the Hamiltonian of (7) reduces to a simple numerical value for the energy (there is no need to diagonalise the Hamiltonian to obtain the spins, we know them already). If there is no external magnetic field, the total energy can thus be found by

$$E = -J \sum_{\langle i,j \rangle} S_z^i S_z^j \tag{9}$$

Now, we can randomly sample each lattice site, and try to flip the spin. After we flip the spin, we calculate the energy again. Intuitively, we know that any natural system is likely to move to a lower energy state. Therefore, if the new energy is less than or equal to the old energy, then we confirm that the spin is flipped, and move on to try and flip the next (random) one. If the new energy is greater than the old one, then the spin is not likely to flip. To do this, we can compute the old energy and new energy in every iteration to find the change in energy. However, that is not very efficient since the energy has to be recomputed twice every iteration, meaning the computer has to add up the spins from all

$N \times N$ lattice site twice. So lets find an expression for ΔE that dosen't require this computation.

We can represent the qubit $S_i = S_{x,y}$ at site $i = (x, y)$ flipping by $S_i \to -S_i$. We know that this will affect only the terms that have S_i as a factor, namely the terms containing its nearest neighbours. We can write out the affected terms as follows

$$f = S_{x,y}S_{x+1,y} + S_{x-1,y}S_{x,y} + S_{x,y}S_{x,y+1} + S_{x,y-1}S_{x,y} \tag{10}$$

Following the spin, f will also flip signs because all the S_i inside has its sign flipped. We can subtract the terms Jf from the energy, which gives the part of the energy that remains unaffected by the qubit flip. Following this, we can add $-Jf$, which is the new part of the energy to the system. Overall, we obtain an expression for the new energy, which we denote by E' as follows

$$E' = -J \sum_{\langle i,j \rangle} S_z^i S_z^j - Jf + (-Jf) = E - 2Jf \tag{11}$$

The change in energy can then be found as

$$\Delta E = E - E' = 2Jf = 2JS_{s,y}(S_{x+1,y} + S_{x-1,y} + S_{x,y+1} + S_{x,y-1}) \tag{12}$$

We can then use the sign of this change in energy to determine whether or not a flip is allowed to occur. Again, a flip that causes $\Delta E \leq 0$ is likely to happen, while a flip that causes $\Delta E > 0$ is not. However, a flip that raises the energy is not impossible. Due to the probabalistic nature of statistical mechanics, the probability of an energy raising flipping can be described by the Metropolis probability, $P = \exp(-\Delta E/k_B T)$. Notice that the flips that raise energy becomes exponentially less likely for higher and higher changes in energies. Notice also, that it is dependent on the temperature T. For low temperatures, the algorithm is less likely to accept flips while for high temperatures, a flip is more easily accepted. In summary, for every flip,

$$\begin{cases} \Delta E \leq 0 & \implies \text{Flip accepted} \\ \Delta E > 0 & \implies P = \exp\left(-\frac{\Delta E}{k_B T}\right) \text{ To accept flip} \end{cases} \tag{13}$$

We can repeat this algorithm many times for random lattice sites until a stable state is reached (a state where no further flips are accepted). For very low temperatures ($T \to 0$), The energy of this state will approach the ground state energy eigenvalue, and this state is the ground state spin configuration that we are looking for.

This algorithm is much faster than doing direct diagonalisation by many orders of magnitudes. However, for larger and larger lattices, convergence will be increasingly slow, and the number of iterations taken will be higher and higher. For perspective, the convergence of a 25×25 lattice took my computer 39640 iterations! In practice, the calculations of actual lattices (which may be millions of sites wide! Since each site represents one single electron) is done on world class supercomputers for months on end!

7 Pauli Matrices

The Pauli matrices used in this set of notes are as follows. We will work in units where $\hbar = 1$.

$$\hat{S}_x = \frac{1}{2}\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$
$$\hat{S}_y = \frac{1}{2}\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$
$$\hat{S}_z = \frac{1}{2}\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

This was chosen for the set of notes because electrons have spin $\pm 1/2$. Note that in the Python program for the 2D Ising model, I have opted for the spin to be simplified to ± 1 to avoid floating point error.