# From Ising Model to Simple Neural Network

Name: 刘文焘
Student Number: 2013301020085
Class: 彭桓武班

## *Abstract*

*The Ising model consists of a large number of very simple units, spin. Although the spins are connected together in a very simple manner, things will become really interesting when we considered the behavior of a large number of spins and allow them to interact. In this case the scientists found that under the appropriate conditions some remarkable things could occur, For example, simulating neural network. This article discusses a simple neural network which based on Ising model, implementing a simple neural network model and analyzing its limitations and shortcoming.*

## I. Introduction

[1] The simplest Ising model assumes an interaction only between nearest neighbors so that the energy of the system is

$$E = -J \sum_{<ij>} s_i s_j \tag{1}$$

, where the sum is over all pairs of nearest neighbor spins <$ij$>, and $J$ is known as the exchange constant, which we will assume to be positive. We can imagine that the spin system is in a particular state (here the E value), corresponding to a particular arrangement of the individual spins. Here, it's clear that the energy of the system is lowest if all of the spins are parallel to one another.

A neural network is composed of a very large number of neurons, which is perhaps the most remarkable object in nature. [2] Biological studies show that the neurons communicate using electrical pulse carried by the axons and dendrites. In some case a high firing rate of the sending neuron will favor a high firing rate of the receiving neuron, in another case a high input firing rate will tend to cause a low firing rate of the receiving neuron. So, we can describe the state of a neuron by its firing rate, which is a function of the firing rate of all of the neurons that have inputs to the neuron.

In this article, a neuron will be modeled as a simple Ising spin. As a spin, which has two possible states, up and down, a neuron will be assumed having only two possible states, firing or not firing. The next section will shows more detail of the neural network model and discusses that how this model can restore information as memory. Section III shows how to simulate this neural network model by Monte

Carlo method. Section IV analyzes the limitations and shortcoming in this model. And Conclusions are presented in Section V.

# II. The Model

In this section we will consider how a neural network can function as a memory. At the beginning, we do some conventions. Firstly, the two possible states of a neuron $n$ will be set to $\pm 1$, where we take $n_i = +1$ to correspond to a firing rate of 1, and $n_i = -1$ to firing rate of 0. Secondly, by associating the firing rate with the value of the neuron we have glossed over the time dependence of the synaptic signals. Thirdly, in this model, we assume that our neurons are permitted an interaction between every pair of neurons. Finally, we assume the connections in this neural network are symmetric.

[3] We can find it very useful to consider the effective energy of this neural network system, which can be written as

$$E = -\sum_{i,j} J_{i,j} n_i n_j \tag{2}$$

, where the sum here is over all pairs of neurons $i$ and $j$ in the network. The $J_{i,j}$ are related to the strengths of the synaptic connections, describing the influence of neuron $i$ on the firing rate of neuron $j$. The sum of the synaptic inputs to neuron $i$ will be

$$E_i = \sum_{j} J_{i,j} n_j \tag{3}$$

, and this will determine the firing rate. If this input is negative, $n_i = -1$; If this input is positive, $n_i = +1$.

In order to make this model function as memory, we need that the neuron directions change with time in such way that the neuron configuration eventually ends up in the desired state to recall a pattern. For example, if we want to recall the number $0$, we want the system to take on the configuration in Figure 1. This time evolution is accomplished using Monte Carlo method, which will be discussed in next section. According to the Monte Carlo method, starting from some particular configuration of the entire network, the system will finally change to the stable configuration we want, where the entire energy in this system is a minimum value.

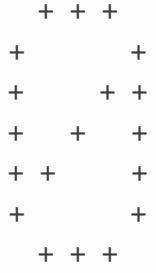

**FIGURE 1**: a $8 \times 5$ lattice of neurons with a particular configuration of firing and not firing. Here the firing neuron is displyed in '+', and not firing neuron is replaced by blanks. This network holds the number $0$.

[4]The interaction energies $J_{i,j}$ are key elements in this model. We let $n_i(m)$ denote the configuration of spin $i$ in pattern $m$. If we want this memory system to have this configuration sa one of its stored patterns, we need choose a particular $J_{i,j}$ so as to make the energy a minima value. Obviously that there is no unique solution for this problem, we could choose

$$J_{i,j} = n_i(m)n_j(m) \tag{4}$$

. We can see that this choice for $J_{i,j}$ will make $n_i(m)$ an energetically stable pattern as follows.

$$E(m) = -\sum_{i,j} J_{i,j}n_i n_j = -\sum_{i,j} n_i(m)n_j(m)n_i n_j \tag{5}$$

. If the network is in pattern $m$, we will have $n_i = n_i(m)$ and $n_j = n_j(m)$, so each term in the sun will be unity, making the energy large and negative. So we can obtain that some similar pattern whose energy is around a desired pattern will be recognized as the pattern we want.

[5]Further more, we need this network model to storing as many patterns as possible. In that case we can choose the $J_{i,j}$ according to

$$J_{i,j} = \frac{1}{M}\sum_{m} n_i(m)n_j(m) \tag{6}$$

, where the index $m$ refers to the stored patterns, and there are a total of $M$ such patterns (we will see in section IV that there is a limit on the total number of patterns that can be stored).

# III. The Simulation

We are now apply the Monte Carlo method to the memory system in this model. The algorithm for the neural network model is given by Algorithm $1$.

*Algorithm $1$. Monte Carlo Method for $N$ neural network*

---

1: Compute all $J_{i,j}(i, j = 1, 2, \ldots, N)$ by equation (6);
2: Set all $n_k(k = 1, 2, \ldots, N)$ which is similar to a particular pattern;
3: Loop through the $N$ neurons of the network, matching this pattern to its real pattern
    a. Calculate $E_i$ by equation (3);

b. If $E_i < 0$, $n_i = -n_i$ change the neuron's state;
4: Display this new pattern.

The important point of Algorithm 1 is that the Monte Carlo changing rule ensure that the system will always evolve in time to state with same or lower energy. If conditions are right, this state corresponds to the pattern that is recalled by its memory.

Here for example, we use a $8 \times 5$ lattice of neurons and we store three numbers $0, 1, 2$ as three patterns on configuration in Figure 2. For each neuron $n_i (i = 1, 2, \ldots, 40)$, if it need to firing in a particular pattern, we set $n_i = 1$ else set $n_i = -1$. Here we use three pattern so the $M$ in equation (6) is equal to 3 and we can compute the $J_{i,j}$ according to this three pattern. Then we choose a pattern in Figure 3(a), which is likely to number $1$. After matching this pattern we can obtain its real pattern is number $1$ in Figure 3(b).

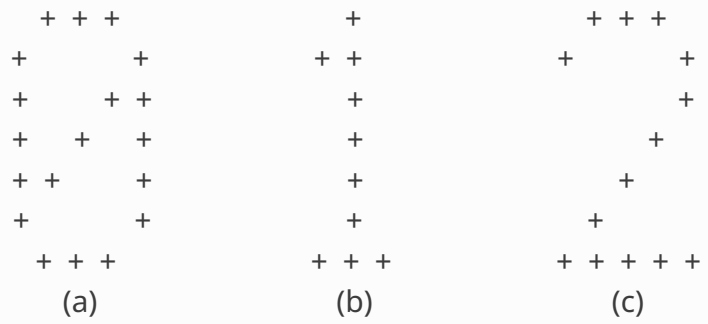


**FIGURE 2**: three $8 \times 5$ lattice of neurons with a particular configuration. This network holds three numbers $0, 1, 2$.

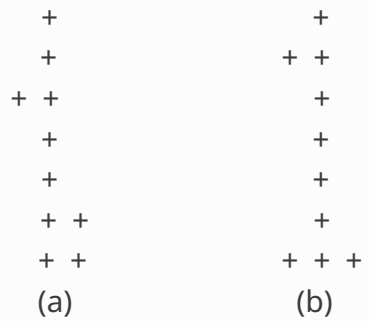


**FIGURE 3**: (a) neuron arrangement close to the number $1$, but with few neurons flipped to the wrong state; (b) neuron configuration after Mento Carlo pass through the lattice.

# IV. The Analysis

From previous section we can see that those neural network system could have a memory capacity. However, this model is very simple and it has many limitations and shortcomings.

In previous section we use a $8 \times 5$ lattice of neurons to store three different patterns. In fact, [6] it has been found that if the number of stored patterns exceeds ~ $0.13N$, all of the stored patterns become unstable. Now we let this memory system to store ten different patterns, number $0\ 9$. In this case, if we set the initial pattern to be the same as previous example, the final pattern is not the number $1$, but an another pattern which is similar with number $$3($*butitisnotthenumber*3$ pattern), which is shown in Figure 4.

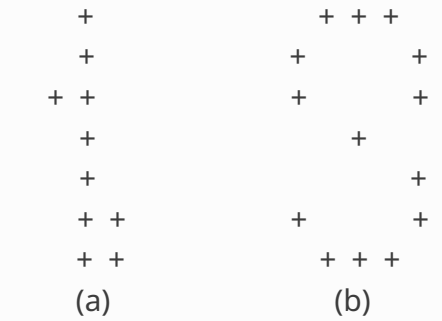


**FIGURE 4**: (a) neuron arrangement close to the number $1$, but with few neurons flipped to the wrong state; (b) a wrong matching pattern after recall the memory.

For now, we only considered the case that the patterns are not rotatable. In real brain neural network, our memory system can distinguish the rotated pattern from our eyes, despite the need to deal with for a long time. In Algorithm $1$, we can not distinguish the pattern which just simplely rotates $180^{\circ}$. For example, in Figure 5, people can easily identify the pattern (a) is a rotated number $2$, while our model apparently get a wrong answer.

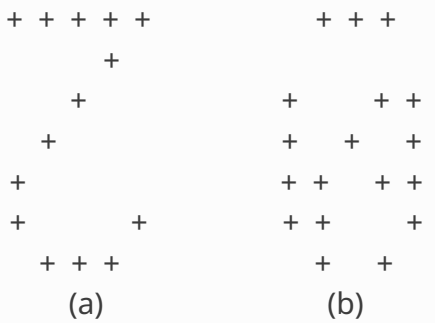

**FIGURE 5**: (a) neuron arrangement rotated $180^{\circ}$ from the number $2$, which people can shortly recognizes; (b) the wrong matching pattern after recall the model's memory.

In fact, we can modify Algorithm $1$ in a simple way to make this model be able to distinguish rotate pattern. However, this modification depends on how the memory system identify a pattern. For example, for our human eye, we receive scenes from our eyes and think of it as a flat image. That is mean our recognition about rotation just comes from rotating the picture by our brain. So from this way, we can modify the Algorithm $1$ to Algorithm $2$ as below. Here simplicity we consider a $N \times N$ lattice ($N$ is large enough to hold a whole pattern), and let $(\frac{N}{2}, \frac{N}{2})$ to be the origin coordinates. all calculation results are integer.

*Algorithm $2$. A Improved Version of Algorithm $1$*

---

1: Compute all $J_{i,j}(i, j = 1, 2, \ldots, N \times N)$ by equation (6);
2: Set all $n_k(k = 1, 2, \ldots, N \times N)$ which is similar to a particular pattern;
3: Set match level $r(r \in [0, 1])$ that can be confident to the final pattern;
4: Set rotation accuracy $\Delta\theta$ for every slight rotation;
5: Loop through the angle from $0$ to $2\pi$ at intervals of $\Delta\theta$
- Rotate all of the $N \times N$ neurons $-i\Delta\theta(i = 0, 1, \ldots, 2\pi/\Delta\theta)$, where the center is origin;
- Loop through the $N \times N$ neurons of the network, matching this pattern to its final pattern
  - Calculate $E_j$ by equation (3);
  - If $E_j < 0$, $n_j = -n_i$ change the neuron's state;
- If the similarity between initial pattern and final pattern is greater than $r$, store the final pattern and exit;
- Else if the similarity is greater than the similarity in previous loop step, store the final pattern and continue;

6: Display this final pattern.

The key element in Algorithm $2$ is how to rotate the pattern. But here we do not make extension, you can find it in any book contains basis of computer graphics.

# V. The Conclusion

In this article I have been able to give only a very simple discussion of a particular neural network based on Ising model. I complement this simple model and then analyse its limitation, confirming that when the total number of patterns is bigger than its $13$ it will be unusable. At the end of this article, I extend this model to be able to distinguish rotate patterns and give a improved algorithm. Here I

would like to thank Cao Yi for his ACII table of $0\ 9$ numbers. His idea gave me much help.

[7] Neural network are now of very great interest to scientists from a number of different fields, as they seem capable of massively parallel processing and could provide an efficient means for solving certain computationally very difficult problems, such as those connected with image processing, machine learning and pattern recognition.

## Reference

[1–7] Nicholas J. Giordano and Hisao Nakanishi, *Computational Physics(Second Edition)*. Beijing: Tsinghua University Press, December 2007. p418-p434