ENS Lyon

Théorie des graphes avancée:

Professeur: Eric Thierry

Hopfield Network

Alice Julien-Laferriere

Lyon

1 Introduction

A neural network (or more formally artificial neural network) is a mathematical model or computational model inspired by the structure and functional aspects of biological neural networks. It consists of an interconnected group of artificial neurons.

The original inspiration for the term Artificial Neural Network came from examination of central nervous systems and their neurons, axons, dendrites and synapses which constitute the processing elements of biological neural networks.

The first model of a neuron was presented in 1943 by W. Mc Culloch and W. Pitts, and in 1958 Rossenblatt conceived the Perceptron. His work had big repercussion but in 1969 a violent critic by Minsky and Papert was published.

The work on neural network was slow down but John Hopfield convinced of the power of neural network came out with his model in 1982 and boost research in this field. Hopfield Network is a particular case of Neural Network. It is based on physics, inspired by spin system.

This work is mainly based on the paper of J.Bruck : *On the convergence properties of Hopfield Model* [1] and the chapter 13 of the book of R.Rojas : *Neural Networks* [2].

2 Hopfield Network Elements

Hopfield network consists of a set of interconnected neurons which update their activation values asynchronously. The activation values are binary, usually {-1,1}. The update of a unit depends on the other units of the network and on itself. A unit i will be influence by an other unit j with a certain weight w_{ij}, and have a threshold value.

So there is a constraint due to the other neurons and due the specific threshold of the unit.

Update and parameters

The new activation value (state) of a neuron is compute, in discret time, by the function :

$$x_i(t+1) = sign(\sum_{j=1}^{n} x_j(t) w_{ij} - \theta_i) \tag{1}$$

or

$$X = sign(XW - T)$$

where X, W, T and the sign function are :

- X is the activation value of the n units/neurons : $X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$
- W is the weight matrix : $W = \begin{pmatrix} w_{11} & w_{12} & \dots & w_{1n} \\ w_{21} & w_{22} & \dots & w_{2n} \\ \vdots & & \ddots & \vdots \\ w_{n1} & \dots & \dots & w_{nn} \end{pmatrix}$ where w_{ij} can be interpreted as the influence of neuron i over neuron j (and reciprocally)
- T is the threshold of each unit : $T = \begin{pmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{pmatrix}$
- the sign function is define as :

$$\begin{cases} +1 & \text{if} \quad x \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

We can easily represent an Hopfield Network by a weighted undirected graph were :
- each unit is a vertex
- the weighted edge between each vertex is the weight, W is then useful as an adjacent

matrix.

For example, the graph 1 will correspond to :

$$X = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}, W = \begin{pmatrix} 0 & -1 & 1 \\ -1 & 0 & -1 \\ 1 & -1 & 0 \end{pmatrix}, T = \begin{pmatrix} -0.5 \\ -0.5 \\ -0.5 \end{pmatrix}.$$

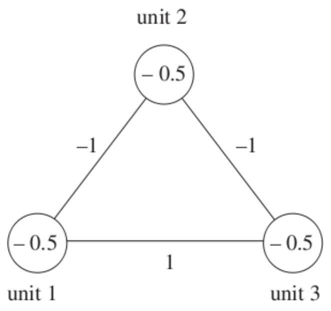

FIGURE 1: Example of a really simple Hopfield network

Usually, an Hopfield Network has a <u>weight matrix symmetric, zero-diagonal</u> (no loop, a unit does not influence on itself). We will only consider that case in our study.

3 Convergence

Hopfield networks converges to a local state. The energy function of a Hopfield network in a certain state is :

$$E_1 := -\frac{1}{2}X^tWX + TX^t = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}x_ix_j + \sum_{i=1}^{n}\theta_i x_i \tag{2}$$

$$E_2 := -\frac{1}{2}X^tWX = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}x_ix_j \tag{3}$$

E1 can be useful as a general energy function. More often, we use E2. E2 is equivalent to E1.
In order to take into account the threshold of the units, we add a artificial unit which is in the activation state 1. This unit is linked to the other with the value of their negative threshold. In that case, the new weight matrix will be

$$W_G = \begin{pmatrix} W & T \\ T^t & 0 \end{pmatrix}$$

Those energy functions are helpful to determinate the actual state of the network.

Proposition 3.0.1 *A Hopfield network with n units and asynchronous dynamics which starts from any given network state, eventually reaches a stable state at local minimum of the energy function.*

Proof: As the dynamic is asynchronous, only one unit is evaluated at each time. If the k^{th} unit is selected, there is two options :

1. k does not change state
2. k changes of state $(-1 \to 1$ or $1 \to -1)$

In the first case, the energy function remains the same, $E(t+1) = E(t)$. Else, the energy function changes, according to the new excitation value $(x'_k)$of the k^{th} unit. The difference of energy of the system is given by :

$$E_1(x) - E_1(x') = (-\sum_{j=1}^{n} w_{kj}x_jx_k + \theta_k x_k) - (-\sum_{j=1}^{n} w_{kj}x_jx'_k + \theta_k x'_k)$$

$$\text{Since } w_{kk} = 0,\ E_1(x) - E_1(x') = -(x_k - x'_k) * (\sum_{j=1}^{n} w_{kj}x_j - \theta_k) \tag{4}$$

The second term of this equation : $(\sum_{j=1}^{n} w_{kj}x_j - \theta_k)$ is the excitation e_k of the k^{th} unit. The unit changed its state, so the excitation has a different sign than x_k and $-x'_k$ (according to (1)). Therefore, (4) is positive, which implies that $E(x) > E(x\prime)$ So the energy will decrease for every change, and since there is only a finite number of possible states, the network should reach a state where the energy cannot decrease more, it will be a stable state.

Of course the stable state is not unique but it is certain that the system will converge. The final state will depend on the input, the initial state of the system.

3.1 Link to graph theory

Theoreme 3.1 *[1] Let G=(V,E) a weighted and undirected graph with W being the matrix of its edges weights. Then the minimal cut problem is equivalent to max Q_g where $X = \{-1, 1\}^n$, and*

$$Q_g := \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}x_ix_j$$

Proof: It is clear than $Q_g = -E_1$. So finding the max of Q_g is equivalent to finding the min of E. Let assign a variable x to every node $i \in V$.

Let W^+ denote the sum of the weights of edges in G with both end points equal to 1, and let W^{--} and W^{+-} denote the corresponding sums of the other two cases.

$$Q_g = (W^{++} + W^{--} - W^{+-})$$

which also can be written as

$$Q_g = (W^{++} + W^{--} + W^{+-}) - 2W^{+-}$$

$$Q_g = \frac{1}{2}(\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}) - 2W^{+-} \tag{5}$$

The first term in (5) is constant (equals the sum of eights of edges in G).

Hence, maximization of Q_g is equivalent to minimization of W^{+-}. This maximization is actually a weight of a cut in G with V_1 being the set of nodes in G that correspond to the units that are equal to 1 and V_2 being the set of nodes in G that correspond to the units that are equal to -1.

The theorem 3.1 does not consider that in Hopfield Network, each unit as a threshold, which determine if a unit gets sufficient impulsion to change state. When we take into account the thresholds of the units of the network, we can also prove that finding a stable state will be equivalent to the minimal cut problem.

Theoreme 3.2 *Let $N = (W, T)$ be a network with W being a n*n symmetric zero diagonal matrix.*

Let G be a weighted graph with (n+1) nodes, with its weight matrix W_G being

$$W_G = \begin{pmatrix} W & T \\ T^t & 0 \end{pmatrix}$$

The problem of finding a state V for which E is a global minimum is equivalent to minimum cut problem on the corresponding graph G.

Proof: The graph G contains $n+1$ nodes. The last node included is connected to the other ones by weighted edges. Those edges has a weight $w_{n+1,i} = \theta_i$.

If the state of the node $n+1$ is constrained to -1,

$$E_1 := \frac{1}{2}\sum_{i=1}^{n+1}\sum_{j=1}^{n+1} w_{ij}x_ix_j = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}x_ix_j - \sum_{i=1}^{n}\theta_i x_i$$

And then we can apply the theorem 3.1.The cut is symmetric, $Q_g(X) = Q_g(-X)$. The minimal cut can also be obtained by interchanging V_1 and V_{-1}. And the additional node can be set to 1.

3.2 Minimum cut problem

Definition A cut C = (S,T) is a partition of V of a graph G = (V, E). The cut-set of a cut C = (S,T) is the set $(u, v) \in E \mid u \in S, v \in T$. If the edges are weighted, the value of the cut is the sum of the weights of the edges in the cut set.

The minimal cut problem is : find the cut of smallest "edge weight". The weight inside the set must be greater than the weights between. A cut is minimum if the size of the cut is not larger than the size of any other cut.

The simplistic way to find the minimal cut in an undirected weighted graph is (from [1]) :

1) Start with a random assignment $V \in \{-1, 1\}^n$
2)Chose a node $i \in \{1...n\}$ randomly
3) Compare the sum of the weights of the edges which belong to the cut and incident at node k with the sum of weight of the other edges which are incident at node k. Move node k to the side of the cut which will result in a decrease in the weight of the cut. If there is equality, place the node k in V_1
4) Go to step 2 ...

Theoreme 3.3 *Let N=(W,T) be a network with T=0 and W be a symmetric zero-diagonal matrix. If N is operating asynchronously then it will always converge to a stable state.*

Proof: As said before, we can consider the operation of N as the running of the algorithm above, for the min cut in N. In each iteration, the value of the cut is non increasing (ties are broken), thus the algorithm will always stop resulting in a cut whose weight is a local minimum.

A small example

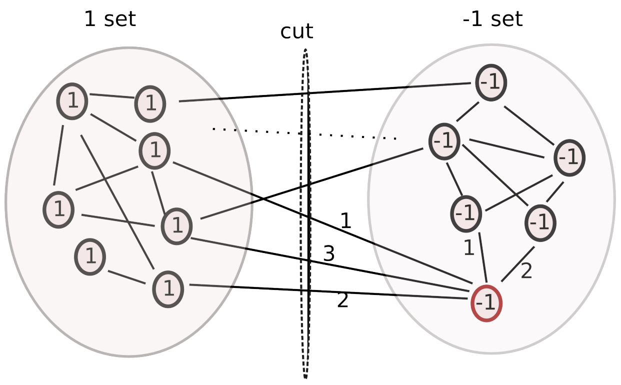

FIGURE 2: A small example

For example, here, for the node circled in red, the outside attraction is higher than the inner attraction (out : 6, in : 3). So the node will change state ($-1 \rightarrow 1$) and the size of the cut will decrease (-3).

4 Application of Hopfield Network :

Neuronal network such as Hopfield Network have diverse applications. Generally they are used for :

– Associative memories : the network is able to memorize some states, patterns.
– Combinatorial optimization : if the problem is correctly modelled , the network can give some minimum and some solution but can rarely find the optimal solution.

Generally Hopfield Network are used as a ≪ black box ≫ to calculate some output resulting from a certain self-organization due to the network. One of the most common application is the Hebbian learning.

4.1 Associative Memory :

4.1.1 Hebbian Learning

In the Hebbian learning, the weight of the matrix results from a unsupervised learning, this learning type is a reinforcement learning. We consider that we can ”impress” patterns to the network in order to recognize one of them even if they are given in input with a lot of noise. The learning phase consists in learning different pattern in order to set the weight of the edges. For each pattern, we can resume the learning phase as :

– if two unit has the same activation state ($\{1, 1\}$ $\{-1, -1\}$), the strength of the connexion is reinforce (w_{ij} increase)
– if two unit has opposite activation state ($\{-1, 1\}$), the strength of the connexion decline.

With those kind of rule, W will store patterns and then allow the network to converge to them. Unfortunately, in that case we can not predict the convergence to the right pattern (the network will still converge but we cannot predict to which stable state).

Learning

The weight matrix is modified according to the m pattern we want to store. Those pattern represent the stable state we hope to reach. The weights are modified, for example :

$$w_{ij}\prime = w_{ij} + \epsilon x_i x_j \qquad (6)$$

(6) is iterated various time for each pattern. x_i and x_j represent the state of the units i and j of the pattern being learned.

ϵ is a constant to prevent the connection to became to big or to small (for example : $\epsilon = \frac{1}{m}$)

When $m = 1$, we always converge to the impress pattern.

When $m > 1$, we can only hope to converge to a known pattern if the different patterns are as orthogonal as possible (the different patterns have to be really different). If they are quite related it is possible to have a mix of different pattern.

It has been shown that Hopfield network can only stock $\simeq 0.14 * n$ patterns (n being the number of units in the network.

4.2 Combinatorial Problem

Hopfield Network can be use for combinatorial problem. If a problem can be written as an isomorphic form of the energy function, the network will be able to find a local minimum of the function but will not guaranty that the solution is optimal.

Usually, in those problems the state of the units are code as $(0, 1)$. We will use this notation, it makes easier the notations.

4.2.1 The travelling salesman

The Travelling Salesman Problem (TSP) is an NP-hard problem in combinatorial optimization. Given a list of cities and their pairwise distances, the task is to find a shortest possible tour that visits each city exactly once. The path has to pass through n cities : S_1, S_2,...S_n. Each city should only be visited once, the salesman has to return to his original point of departure and the length of the path has to be minimal.

The network can be represented as :

	1	2	...	n
S_1				
S_2				
$\vdots$				
S_n				

Where rows are the city and columns are the time. The units are the entry of the matrix. We suppose that the (n+1) column is the first one to keep the cycle. Also only one 1 should appears for each row and column because the salesman only visit a city once and cannot visit two city during the same time step.

The distance between the city S_i and the city S_j is d_{ij}. To find the shortest path, the network will minimize the function :

$$E_{tsp} = \underbrace{\frac{1}{2}\sum_{i,j,k}^{n} d_{ij}x_{ik}x_{j,k+1}}_{L} + \underbrace{\frac{\gamma}{2}(\sum_{j=1}^{n}(\sum_{i=1}^{n} x_{ij} - 1))^2 + \sum_{i=1}^{n}(\sum_{j=1}^{n} x_{ij} - 1)^2))}_{Condition} \tag{7}$$

The first part of the equation, L, is the total length of the trip. $x_{i,k} = 1$ means that the city S_i was visited at time k(otherwise $x_{i,k} = 0$), so if $x_{ik} = 1$ and $x_{i,k+1} = 1$ (S_i visited at time k and S_j visited at time k+1), the distance will be added to the length L.

The second part represent the condition : $\underline{\text{one time in each city}}$. $\sum_{i=1}^{n} x_{ij} - 1)^2$ will be minimal (=0) if for every i (here city), the sum over j (here time) is 1, meaning that a city can only be visited once. $\sum_{j=1}^{n}(\sum_{i=1}^{n} x_{ij} - 1)^2$ will be minimal if for every time (k=1...n), the salesman only visit one city.

γ is a parameter. Unfortunately, there is no other choice than using ≪ trial and error ≫ to determine the value we want to set. If γ is small, the condition will not be respected. If γ is really big, the system will be constraint a lot and L will be hidden (so the minimization of E_{tsp} will not consider the distance between cities).

The weights between units becomes :$w_{ik,jk+1} = -d_{ij} + t_{ik,jk+1}$ where

$$t_{ik,jk+1} = \begin{cases} -\gamma & \text{if the units belong to the same column or row} \\ 0 & \text{otherwise} \end{cases}$$

and the threshold should be set to $-\frac{\gamma}{2}$.

The network will not be able to find the good path. But still, the solutions are approximation. If the number of cities is important (> 100), there is several minima so we cannot prove if the solution given by the system is a good approximation.

4.2.2 Discussion about efficiency

When we try to resolve combinatorial problem, Hopfield network can help if we are able to express the problem as an energy function (and the aim will be minimizing this function).

However, as said before, the optimality is not guaranteed (and really often minimization will failed) and the system could ≪ fall ≫ into a local minima. But this kind of network still remains useful because if a unit is considered as a small processor, neural networks can offer high power of computation and parallelism (thanks to the asynchronous update of the network).

It is also possible to improve the result by adding *noise* to the network dynamics, and hope that it will allow the system to jump the energy ladder.

Finally, an other solution is to increase the possible solutions, meaning that instead of using a discrete network, we can consider a continuous one.

5 The continuous model

In the continuous model, the different states of the units can take any real value between $[0, 1]$.

The dynamic remains asynchronous and the activation function is a sigmoid.

$$x_i = s(u_i) = \frac{1}{1 + e^{-u_i}} \tag{8}$$

For each unit, x_i once the excitation u_i changes slowly as :

$$\frac{du_i}{dt} = \gamma(-u_i + \sum_{j=1}^{n} w_{ij} s(u_j))$$

and the new energy function is

$$E = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} x_i x_j + \sum_{i=1}^{n} \int_0^{x_i} s^{-1}(x) dx$$

The convergence can be prove (for more information : [2])

6 Conclusion

Hopfiel network was a breakthrough in neural network and gave a important dynamism to neural network's research. A lot of result are available and nowadays neural system are used in computer science. Their ability to learn by example makes them very flexible. They are also very well suited for real time systems because of their parallel architecture.

Still, there are also critics. A. K. Dewdney wrote in 1997 *Although neural nets do solve a few toy problems, their powers of computation are so limited that I am surprised anyone takes them seriously as a general problem-solving tool.* Of course, a big problem is that neural network needs a lot of training to be efficient.

Nevertheless, they are used in many different ways : image processing, pattern matching, data deconvolution....

Appendix

Hopfield Application :

This list is not exhaustive and more applications and the references can be found in *Artificial Neural Systems* of Patrick K.Simpson([3]) :

Combinatorial Optimization Applications
Image processing
Signal processing
Graph Coloring Coloring, Graph Flow and Graph Manipulation
ANS programming
Data deconvolution
Pattern Matching
Solving equations and Optimizing functions
Traveling Salesman, Scheduling and Resource Allocation

References

[1] Jehoshua Bruck. On the convergence properties of the hopfield model. *Procceedings of the IEEE*, 78(10), October 1990.

[2] Raul Rojas. *Neural Networks*. Springer, 1996.

[3] Patrick K.Simpson. *Artificial Neural Systems*. Pergamon Press, 1990.

[4] David Kriesel. *A Brief Introduction to Neural Networks*. 2007. chap 8 pp. 125-135, available at http://www.dkriesel.com.

[5] Wikipédia. http://fr.wikipedia.org/wiki/R%C3%A9seau_de_neurones_de_Hopfield.