Neural Networks

Lecture 9

Hopfield Networks (2)

Md. Mijanur Rahman, Prof. Dr.

Dept. of Computer Science and Engineering, Jatiya Kabi Kazi Nazrul Islam University, Bangladesh.

Web: www.mijanrahman.com | Email: mijanjkkniu@gmail.com; mijan@jkkniu.edu.bd

Chapter: Hopfield Networks

- **This chapter covers the following topics:**
  - Introduction to Hopfield Networks
  - Basic Hopfield Net and How It Works
  - Updating Rule in Hopfield Network
  - Discrete Hopfield Network
    - Algorithm and Examples
  - Continuous Hopfield Network
    - Algorithm and Examples
  - Applications

Discrete Hopfield Network…

- A Hopfield network which operates in a discrete line fashion or in other words, it can be said the input and output patterns are discrete vector, which can be generally of two types in nature:
  - Binary [0, 1]
  - Bipolar [-1, +1]
- The weights associated with this network is symmetric with no self-connections and has the following properties.
  1. $w_{ij} = w_{ji}$
  2. $w_{ii} = 0$

Discrete Hopfield Network…

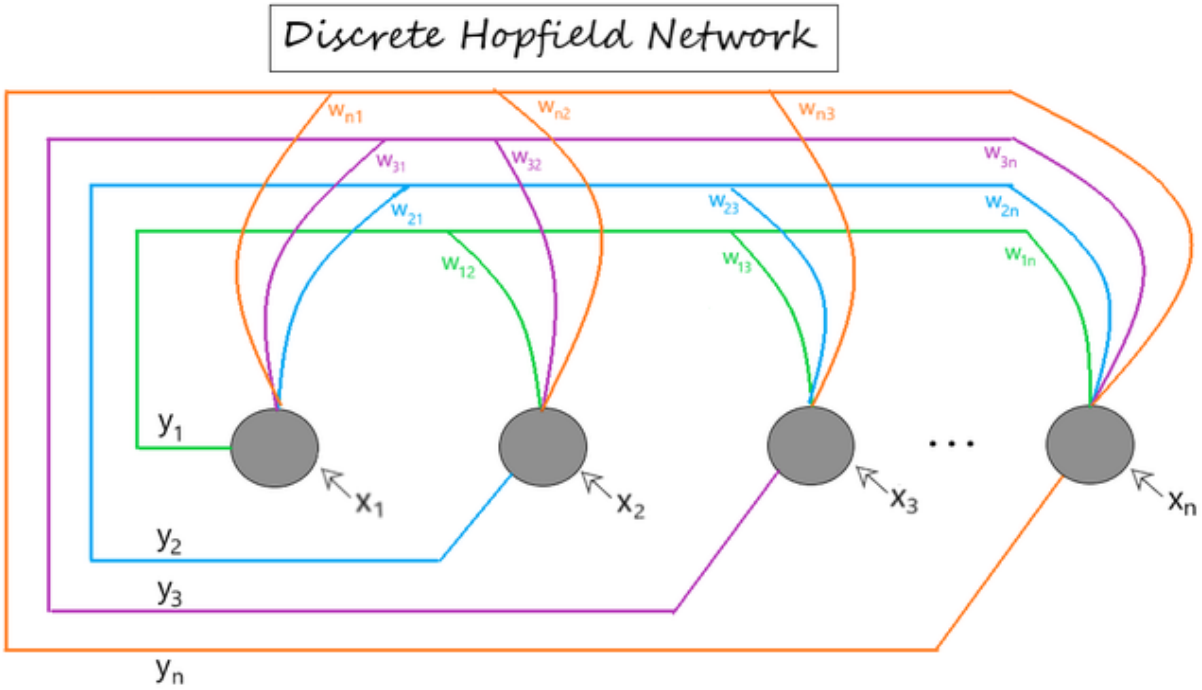
Architecture:

- Following are some important points to keep in mind about discrete Hopfield network −
  - This model consists of neurons with **one inverting and one non-inverting output.**
  - The output of each neuron should **be the input of other neurons but not the input of self.**
  - Weight/connection strength is represented by w_{ij}.
  - Connections can be **excitatory as well as inhibitory**. It would be excitatory, if the output of the neuron is same as the input, otherwise inhibitory.
  - **Weights should be symmetrical**, i.e. $w_{ij} = w_{ji}$

Discrete Hopfield Network…

Architecture:

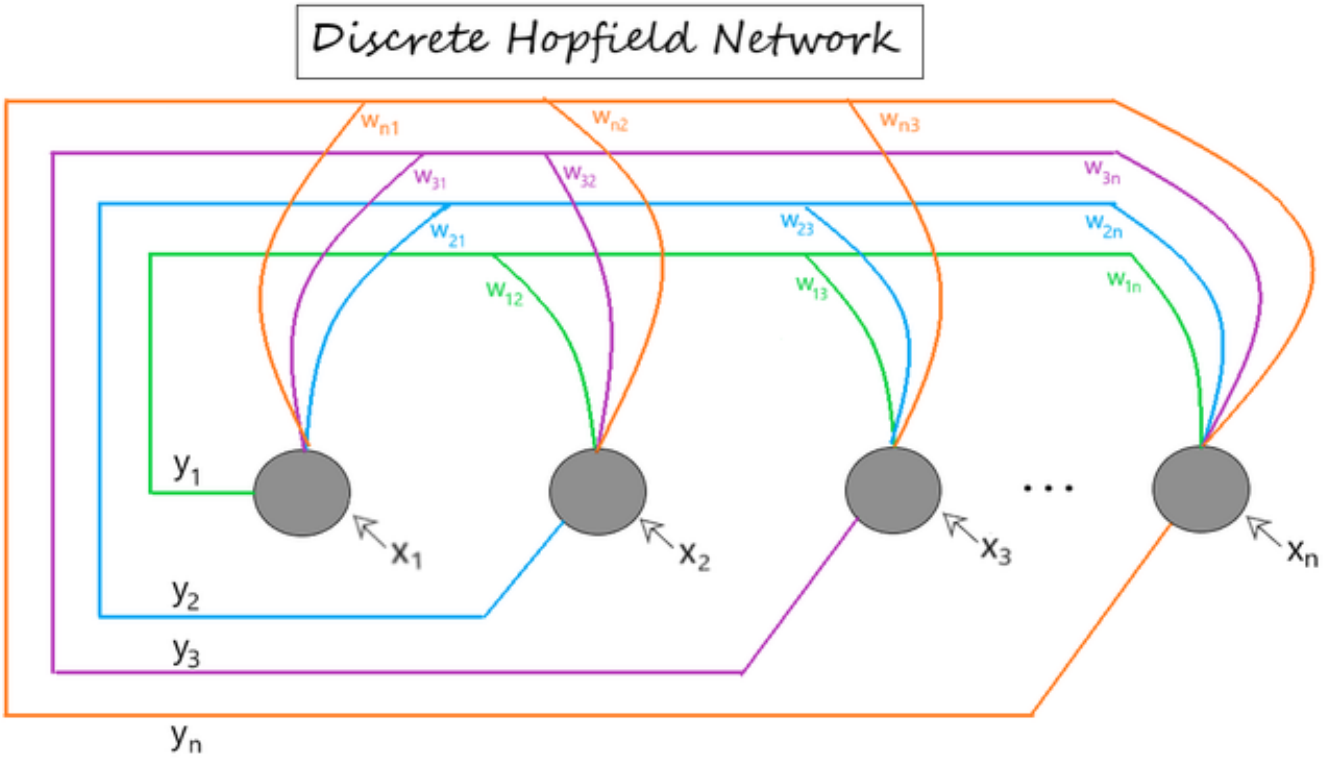
- $[x_1, x_2, \ldots, x_n]$ → Input to the n given neurons.
- $[y_1, y_2, \ldots, y_n]$ → Output obtained from the n given neurons
- W_{ij} → weight associated with the connection between the i^{th} and the j^{th} neuron.



Discrete Hopfield Network…

Architecture:

- The output from $\mathbf{Y_1}$ going to $\mathbf{Y_2}$, $\mathbf{Y_i}$ and $\mathbf{Y_n}$ have the weights $\mathbf{w_{12}}$, $\mathbf{w_{1i}}$ and $\mathbf{w_{1n}}$ respectively. Similarly, other arcs have the weights on them.



Discrete Hopfield Network…

Training Algorithm:

- During the training of the discrete Hopfield network, weights will be updated. As we know, we can have **both binary and bipolar input vectors**.
- **In the case of Binary Input Patterns**, weight updates can be done with the following relation:

Case 1 – Binary input patterns

For a set of binary patterns $s(p)$, p = 1 to P

Here, $s(p) = s_1(p), s_2(p), \ldots, s_i(p), \ldots, s_n(p)$

Weight Matrix is given by

$$w_{ij} = \sum_{p=1}^{P} [2s_i(p) - 1][2s_j(p) - 1] \quad for\ i \neq j$$

Discrete Hopfield Network…

Training Algorithm:

- **In the case of Bipolar Input Patterns**, weight updates can be done with the following relation:

Case 2 – Bipolar input patterns

For a set of binary patterns **s** p , **p = 1 to P**

Here, **s** p = **s**$_1$ p , **s**$_2$ p ,..., **s**$_i$ p ,..., **s**$_n$ p

Weight Matrix is given by

$$w_{ij} = \sum_{p=1}^{P} [s_i(p)][s_j(p)] \quad for\, i \neq j$$

Discrete Hopfield Network…

Testing Algorithm:

Step 1 − Initialize the weights, which are obtained from training algorithm by using Hebbian principle.

Step 2 − Perform steps 3-9, if the activations of the network is not consolidated.

Step 3 − For each input vector **X**, perform steps 4-8.

Step 4 − Make initial activation of the network equal to the external input vector **X** as follows −

$$y_i = x_i \quad for\, i = 1\, to\, n$$

Step 5 − For each unit $\mathbf{Y_i}$, perform steps 6-9.

Step 6 − Calculate the net input of the network as follows −

$$y_{ini} = x_i + \sum_j y_j w_{ji}$$

Discrete Hopfield Network…

Testing Algorithm:

Step 7 − Apply the activation as follows over the net input to calculate the output −

$$y_i = \begin{cases} 1 & if\ y_{ini} > \theta_i \\ y_i & if\ y_{ini} = \theta_i \\ 0 & if\ y_{ini} < \theta_i \end{cases}$$

Here θ_i is the threshold.

Step 8 − Broadcast this output $\mathbf{y_i}$ to all other units.

Step 9 − Test the network for conjunction.

Discrete Hopfield Network…

Energy Function Evaluation:

- The Hopfield networks have an energy function associated with them. It either diminishes or remains unchanged after each iteration (feedback).
- An energy function is defined as a function that is bounded and a non-increasing function of the state of the system.
- Energy function $\mathbf{E_f}$, also called **Lyapunov function,** determines the stability of the discrete Hopfield network, and is characterized as follows:

$$E_f = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j w_{ij} - \sum_{i=1}^{n} x_i y_i + \sum_{i=1}^{n} \theta_i y_i$$

Discrete Hopfield Network…

Energy Function Evaluation:

- **Condition** − In a stable network, whenever the state of node changes, the above energy function will decrease.
- Suppose when node i has changed state from $y_i^{(k)}$ to $y_i^{(k+1)}$, then the energy change ΔE_f is given by the following relation:

$$\Delta E_f = E_f(y_i^{(k+1)}) - E_f(y_i^{(k)})$$

$$= -\left(\sum_{j=1}^{n} w_{ij} y_i^{(k)} + x_i - \theta_i\right)(y_i^{(k+1)} - y_i^{(k)})$$

$$= -(net_i)\Delta y_i$$

Here $\Delta y_i = y_i^{(k+1)} - y_i^{(k)}$

- The change in energy depends on the fact that only one unit can update its activation at a time.

Discrete Hopfield Network…

Example Problem: Consider the following problem.

- We are required to create a Discrete Hopfield Network with bipolar representation of input vector as [1 1 1 -1] or [1 1 1 0] (in case of binary representation) is stored in the network.

- Test the Hopfield network with missing entries in the first and second components of the stored vector (i.e., [0 0 1 0]).

Discrete Hopfield Network…

Step-by-Step Solution:

Step 1 - Given input vector, x = [1 1 1 -1] (bipolar), and we initialize the weight matrix ($w_{ij)}$ as:

$$w_{ij} = \sum [s^T(p)t(p)]$$

$$= \begin{bmatrix} 1 \\ 1 \\ 1 \\ -1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & -1 \\ -1 & -1 & -1 & 1 \end{bmatrix}$$

Step 2 – Weight matrix with no self-connection is given by:

$$w_{ij} = \begin{bmatrix} 0 & 1 & 1 & -1 \\ 1 & 0 & 1 & -1 \\ 1 & 1 & 0 & -1 \\ -1 & -1 & -1 & 0 \end{bmatrix}$$

Discrete Hopfield Network…

Step-by-Step Solution:

Step 3 - As per the question, input vector x with missing entries,

x = [0 0 1 0] ([x1 x2 x3 x4]) (binary)

- Make y_i = x = [0 0 1 0] ([y1 y2 y3 y4])

Step 4 - Choose unit y_i (order doesn't matter) to update its activation.

- Take the i^{th} column of the weight matrix for calculation.

(We will do the next steps for all values of y_i and check if there is convergence or not.)

Discrete Hopfield Network…

Step-by-Step Solution:

- **Step 4 –**

$$y_{in_1} = x_1 + \sum_{j=1}^{4}[y_j w_{j1}]$$

$$= 0 + \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \\ -1 \end{bmatrix}$$

$$= 0 + 1$$

$$= 1$$

Applying activation, $y_{in_1} > 0 \implies y_1 = 1$

giving feedback to other units, we get

$y = \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}$

which is not equal to input vector

$x = \begin{bmatrix} 1 & 1 & 1 & 0 \end{bmatrix}$

Hence, no covergence.

For the next unit, we will use the updated value from feedback.

(i.e. y = [1 0 1 0])

Discrete Hopfield Network…

Step-by-Step Solution:

- **Step 4 –**

$$y_{in_3} = x_3 + \sum_{j=1}^{4}[y_j w_{j3}]$$

$$= 1 + \begin{bmatrix}1 & 0 & 1 & 0\end{bmatrix} \begin{bmatrix}1\\1\\0\\-1\end{bmatrix}$$

$$= 1 + 1$$

$$= 2$$

Applying activation, $y_{in_3} > 0 \implies y_3 = 1$

giving feedback to other units, we get

$y = \begin{bmatrix}1 & 0 & 1 & 0\end{bmatrix}$

which is not equal to input vector

$x = \begin{bmatrix}1 & 1 & 1 & 0\end{bmatrix}$

Hence, no covergence.

Now for next unit, we will take updated value via feedback.

(i.e. y = [1 0 1 0])

Discrete Hopfield Network

Step-by-Step Solution:

- **Step 4 –**

$$y_{in_2} = x_2 + \sum_{j=1}^{4}[y_j w_{j2}]$$

$$= 0 + \begin{bmatrix}1 & 0 & 1 & 0\end{bmatrix} \begin{bmatrix}1\\0\\1\\-1\end{bmatrix}$$

$$= 0 + 1 + 1$$

$$= 2$$

Applying activation, $y_{in_2} > 0 \implies y_2 = 1$

giving feedback to other units, we get

$$y = \begin{bmatrix}1 & 1 & 1 & 0\end{bmatrix}$$

which is equal to input vector

$$x = \begin{bmatrix}1 & 1 & 1 & 0\end{bmatrix}$$

Hence, covergence with vector x.

Continuous Hopfield Network…

- In comparison with the Discrete Hopfield network, the continuous network has time as a continuous variable.
- So, instead of getting binary/bipolar outputs, we can obtain values that lie between 0 and 1. It can be used to solve constrained optimization and associative memory problems.
- The output is defined as:

$v_i = f(u_i)$

where, v_i = output from the continuous Hopfield network.

u_i = internal activity of a node in a continuous Hopfield network.

- It is also used in auto association and optimization problems, such as travelling salesman problem.

Continuous Hopfield Network…

Key Idea:

- Neuron output is continuous:

$$v_i \in [0,1] \quad \text{or} \quad (-1,1)$$

- Uses a nonlinear activation function (typically sigmoid).
- Dynamics evolve over continuous time

Mathematical Model:

- Each neuron has: Internal state, u_i, and output, $v_i = f(u_i)$.
- Activation Function (Sigmoid):

$$v_i = \frac{1}{1 + e^{-u_i}}$$

Continuous Hopfield Network…

Dynamic Update Equation:

- The system evolves according to: $$\frac{du_i}{dt} = -u_i + \sum_{j=1}^{N} w_{ij} v_j + I_i$$

- Where: u_i : internal potential
  v_i : output
  w_{ij} : weight
  I_i : external input

Energy Function:

$$E = -\frac{1}{2} \sum_i \sum_j w_{ij} v_i v_j + \sum_i \int_0^{v_i} f^{-1}(x)\, dx - \sum_i I_i v_i$$

- The network evolves to minimize this energy, reaching a stable equilibrium.

Continuous Hopfield Network…

Algorithm (Step-by-Step):

- Step 1: Initialize
  - Choose the number of neurons N.
  - Initialize: Weight, w_{ij} ; Inputs, ; I_i; Initial State, $u_i(0)$.

- Step 2: Compute Outputs

$$v_i = f(u_i)$$

- Step 3: Update Internal States

$$u_i(t + \Delta t) = u_i(t) + \Delta t \left(-u_i + \sum_j w_{ij} v_j + I_i \right)$$

Continuous Hopfield Network…

Algorithm (Step-by-Step):

- Step 4: Iterate
  - Repeat Steps 2–3 until:
    - u_i converges (no significant change).
- Step 5: Output
  - Final v_i gives solution (often optimization result).

Continuous Hopfield Network…

Algorithm: Numerical Example (Simple)

- Problem Setup:
  - 2 neurons
  - Weight matrix:

$$W = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

  - No external input: $I_i = 0$

- Initial State:

$$u_1 = 0.5, \quad u_2 = -0.5$$

Continuous Hopfield Network…

Algorithm: Numerical Example (Simple)

- Step 1: Compute Outputs

$$v_1 = \frac{1}{1 + e^{-0.5}} \approx 0.62$$

$$v_2 = \frac{1}{1 + e^{0.5}} \approx 0.38$$

- Step 2: Update u_1

$$\frac{du_1}{dt} = -0.5 + (1)(0.38) = -0.12$$

$$u_1^{new} = 0.5 + \Delta t(-0.12)$$

If $\Delta t = 1$:

$$u_1 = 0.38$$

Continuous Hopfield Network

Algorithm: Numerical Example (Simple)

- Step 3: Update u_2

$$\frac{du_2}{dt} = -(-0.5) + (1)(0.62) = 0.5 + 0.62 = 1.12$$

$$u_2 = -0.5 + 1.12 = 0.62$$

- Step 4: Next Iteration

Now:

$$u = [0.38, 0.62]$$

Recompute:

$$v_1 \approx 0.59, \quad v_2 \approx 0.65$$

- Continue until values stabilize.

Applications

- **Applications of Discrete Hopfield Network (DHN):**
  - Associative memory (recall patterns from noisy input)
  - Pattern recognition (e.g., character recognition)
  - Image restoration (noise removal)
  - Error correction in communication systems
  - Simple optimization problems
- **Applications of Continuous Hopfield Network (CHN):**
  - Optimization problems (e.g., Traveling Salesman Problem)
  - Combinatorial optimization (scheduling, assignment)
  - Image processing (smoothing, segmentation)
  - Signal processing (noise filtering)
  - Resource allocation problems

HOPFIELD NETWORKS

The End.