HOPFIELD NETWORK

INT301 Bio-computation, Week 14, 2021

The Hopfield Network

- In 1982, Hopfield, a Caltech physicist, mathematically tied together many of the ideas from previous research.
- A fully connected, symmetrically weighted network where each node functions both as input and output node.
- Used for
  - Associated memories
  - Combinatorial optimization
- Major contribution of John Hopfield to NN
  - Treating a network as a dynamic system
  - Introduced the notion of energy function and attractors into NN research

The Hopfield Network

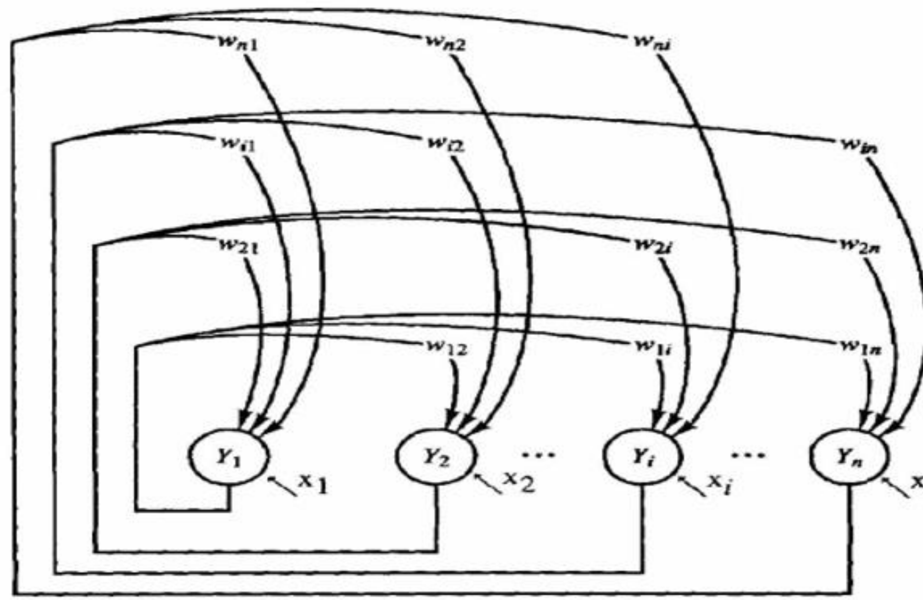
- Different forms: discrete & continuous
- We will focus on the **discrete** Hopfield model, because its mathematical description is more straightforward.
- In the discrete model, the output of each neuron is either 1 or –1.
- In its simplest form, the output function is the **sign function**, which yields 1 for arguments ≥ 0 and –1 otherwise.

Discrete Hopfield Network

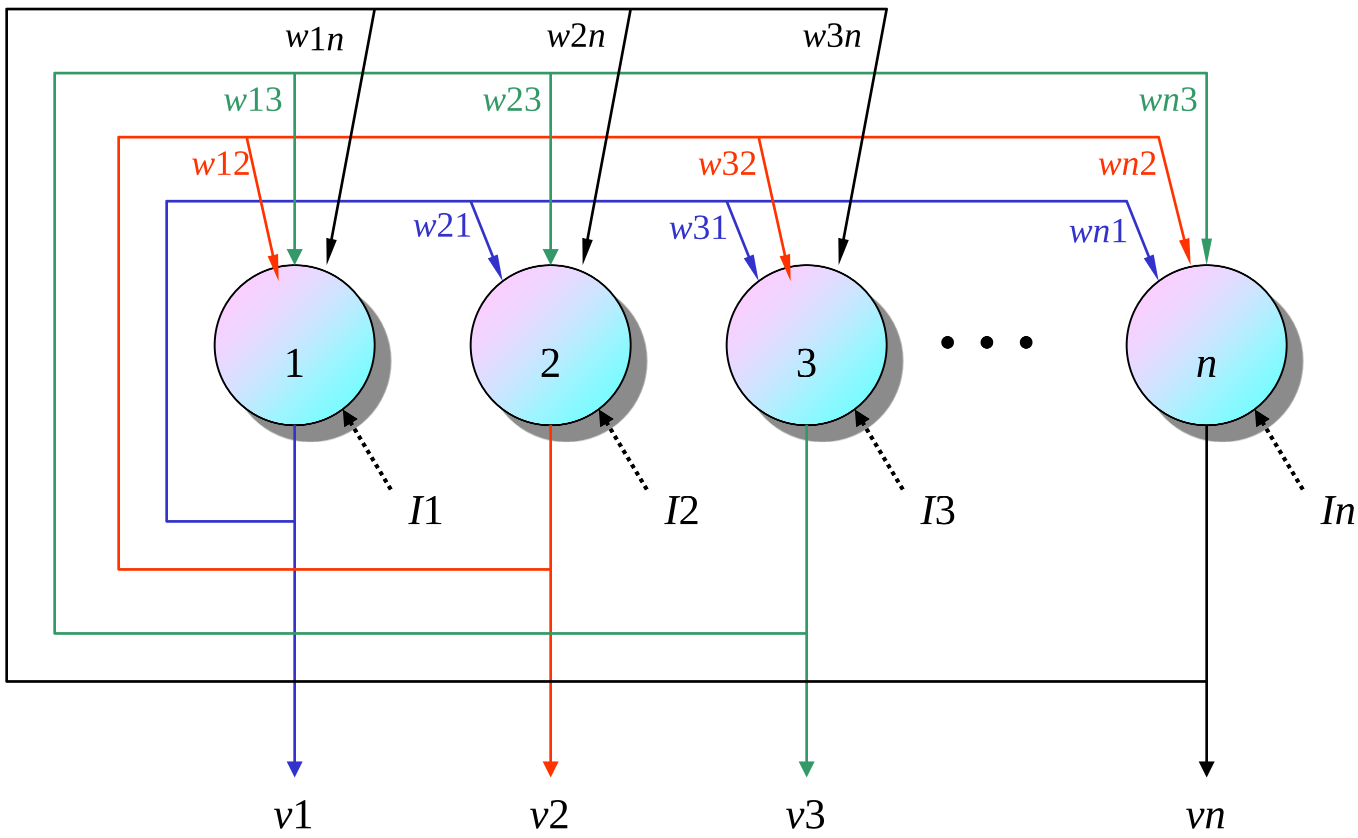
- **Architecture:**
  - single layer (units serve as both input and output)
  - nodes are threshold units (binary or bipolar)
  - weights: fully connected, symmetric, and zero diagonal

$$w_{ij} = w_{ji}$$
$$w_{ii} = 0$$

  - x_i are external inputs, which may be transient or permanent



Discrete Hopfield Network



$$w_{ij} = w_{ji}$$

$$w_{ii} = 0$$

Discrete Hopfield Network

- Storage is performed according to the following equation:

$$w_{ij} = \frac{1}{N} \sum_{p=1}^{P} x_i^p x_j^p$$

- The weight matrix is symmetrical, i.e., $w_{ij} = w_{ji}$.

- The constraint condition $w_{ii} = 0$ is important for the network behavior. It can be mathematically proven that *under these conditions the network will reach a stable activation state within an infinite number of iterations.*

Discrete Hopfield Network

- In the discrete version of the model, each component of an input or output vector can only assume the values 1 or –1.
- The output of a neuron i at time t is then computed according to the following formula:

$$v_i(t) = \operatorname{sgn}\left(\sum_{j=1}^{N} w_{ij} v_j (t-1) \right)$$

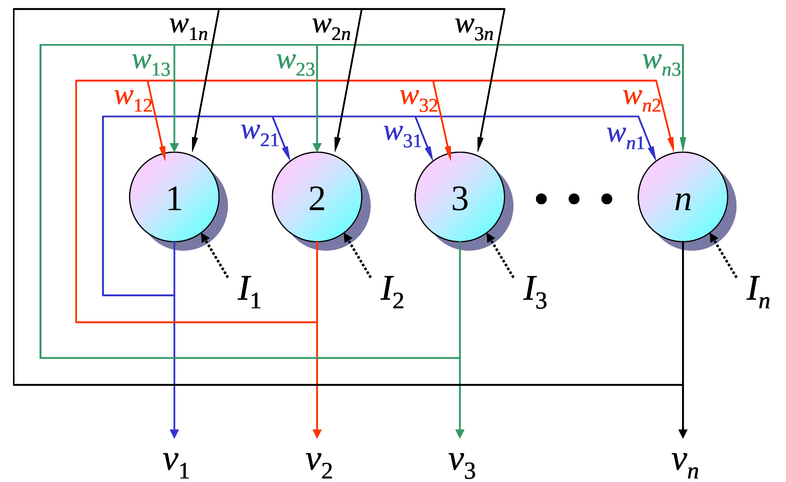
- This recursion can be performed over and over again.

Discrete Hopfield Network

- What does such a stable state look like?
- The network associates input patterns with themselves, which means that in each iteration, the activation pattern will be drawn towards one of those patterns.
- After converging, the network will most likely present one of the patterns that it was initialized with.
- Therefore, Hopfield networks can be used to restore incomplete or noisy input patterns.

Discrete Hopfield Network

- Recall
  - Use an input vector to recall a stored vector
  - Each time, randomly select a unit for update
  - Periodically check for convergence (stable state)
- Asynchronous mode update rule



Stable?

$$H_i(t+1) = \sum_{\substack{j=1 \\ j \neq i}}^{n} w_{ij} v_j(t) + I_i$$

$$v_i(t+1) = \operatorname{sgn}\left[H_i(t+1)\right] = \begin{cases} 1 & H_i(t+1) \geq 0 \\ -1 & H_i(t+1) < 0 \end{cases}$$

- **Example:**
  - A 4 node network, stores 2 patterns (1 1 1 1) and (-1 -1 -1 -1)
  - Weights: $w_{\ell,j} = 1$, for $\ell \neq j$, and $w_{j,j} = 0$ for all j
- Corrupted input pattern: (1 1 1 -1)

Node selection		**output pattern**
node 2:	$w_{2,1}x_1 + w_{2,3}x_3 + w_{2,4}x_4 + I_2 = 1 + 1 - 1 + 1 = 2$	(1 1 1 -1)
node 4:	$1+1+1-1 = 2$	(1 1 1 1)

No more change of state will occur, the correct pattern is recovered

- Equal distance: (1 1 -1 -1)

node 2:	net = 0, no change	(1 1 -1 -1)
node 3:	net = 0, change state from -1 to 1	(1 1 1 -1)
node 4:	net = 0, change state from -1 to 1	(1 1 1 1)

No more change of state will occur, the correct pattern is recovered

If a different node selection order is used, the stored pattern (-1 -1 -1 -1) may be recalled

- Missing input element: (1 0 -1 -1)

Node selection	**output pattern**
node 2: $w_{12}x_1+w_{32}x_3+w_{42}x_4+I_2=1-1-1+0<0$	(1 -1 -1 -1)
node 1: net = -3, change state to -1	(-1 -1 -1 -1)

No more change of state will occur, the correct pattern is recovered

- Missing input element: (0 0 0 -1)

  the correct pattern (-1 -1 -1 -1) is recovered

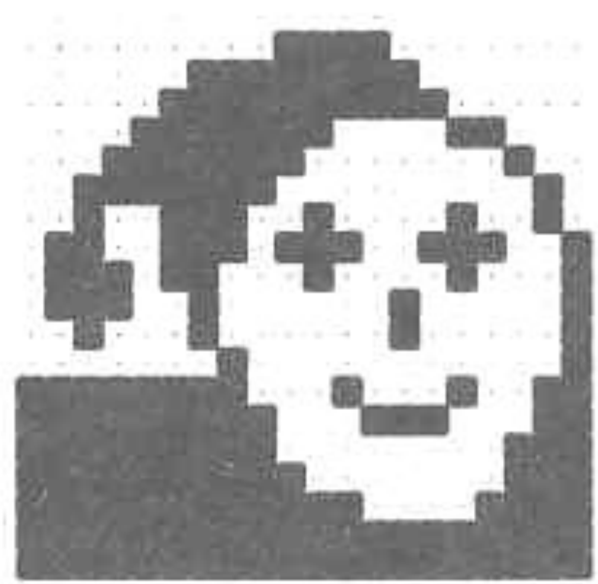
  This is because the AM has only 2 attractors

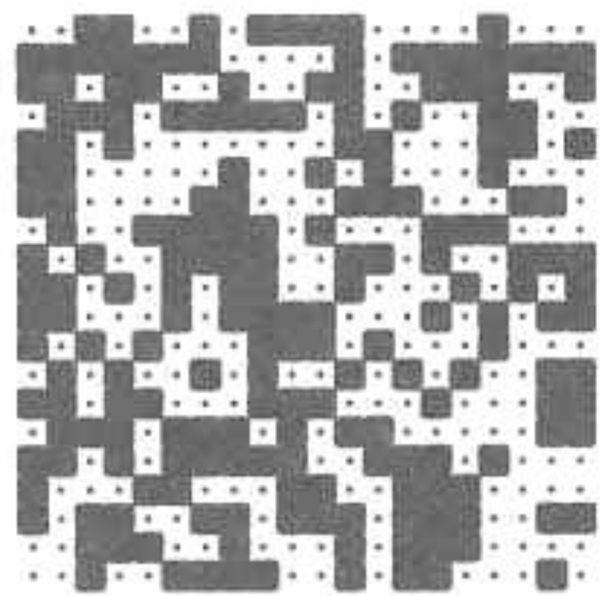
  (1 1 1 1) and (-1 -1 -1 -1)

  When spurious attractors exist (with more memories), pattern completion may be incorrect

Example

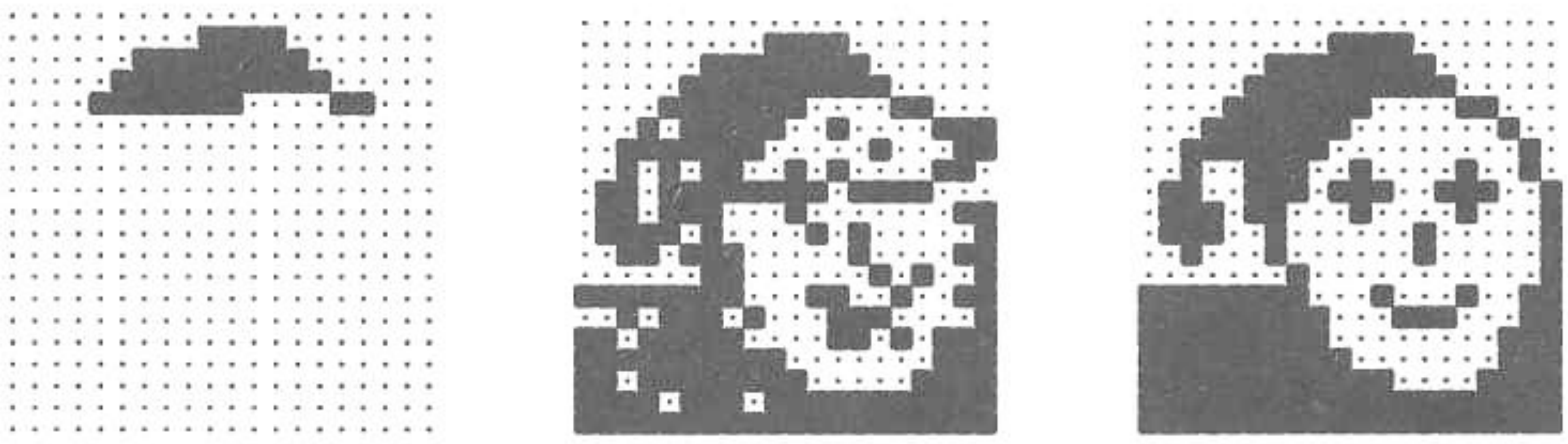
- Image reconstruction (Ritter, Schulten, Martinetz 1990)
- A 20×20 discrete Hopfield network was trained with 20 input patterns, including the one shown in the left figure and 19 random patterns as the one on the right.





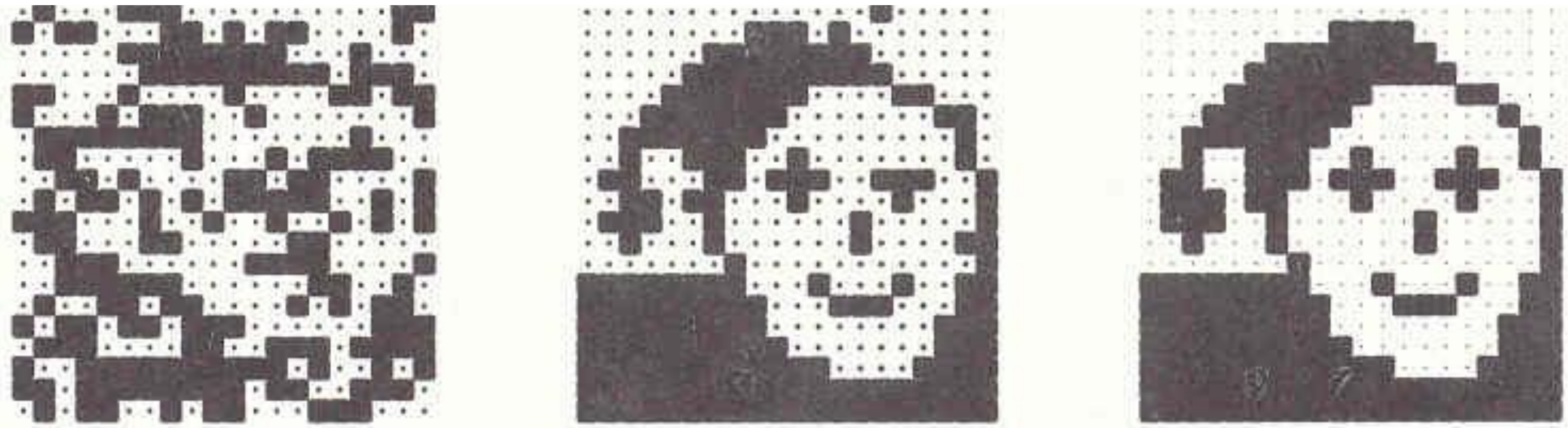
Example

- After providing only one fourth of the “face” image as initial input, the network is able to perfectly reconstruct that image within only two iterations.



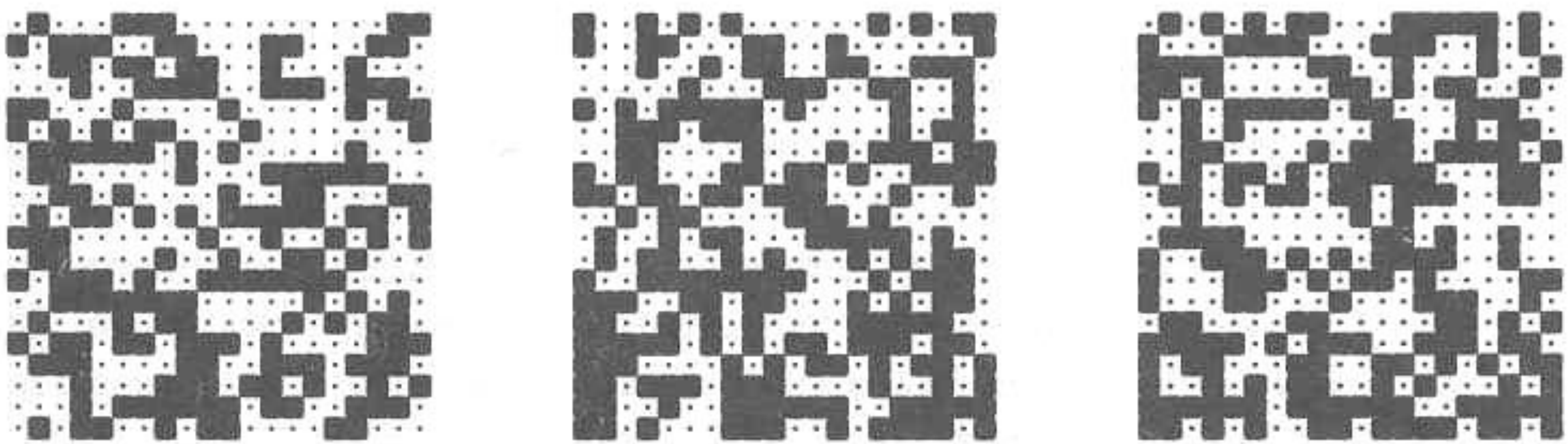
Example

- Adding noise by changing each pixel with a probability p = 0.3 does not impair the network's performance.
- After two steps the image is perfectly reconstructed.



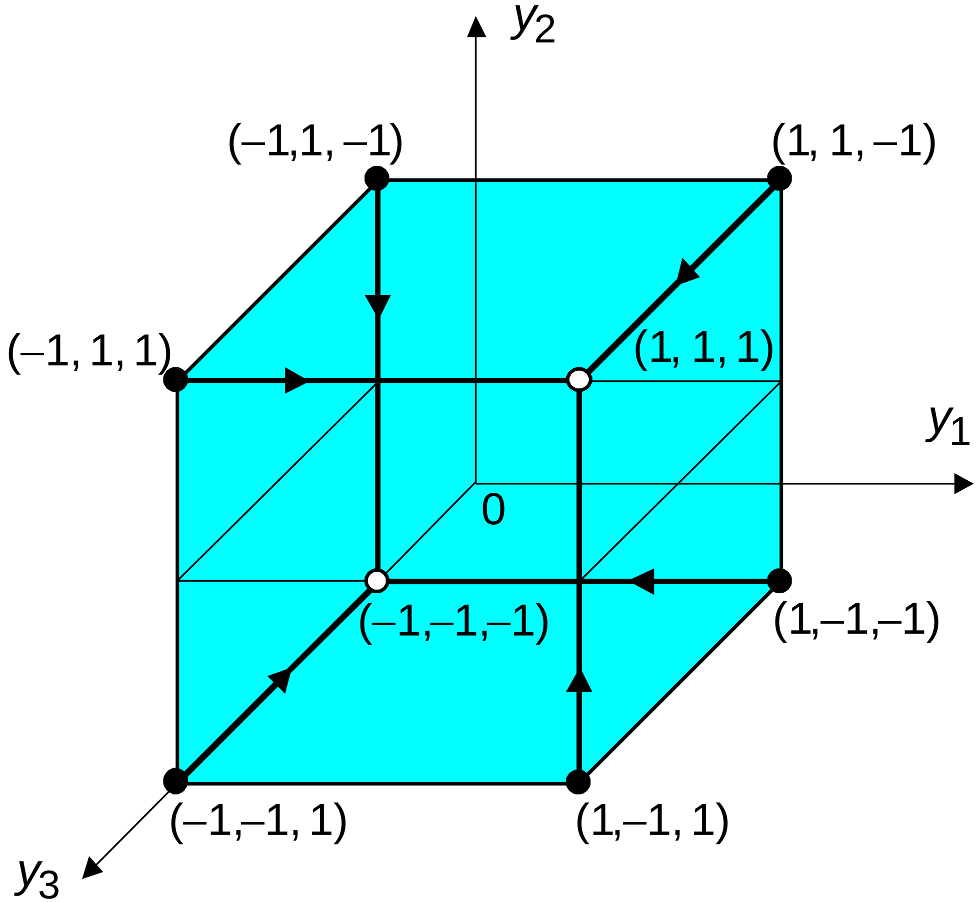
Example

- However, for noise created by p = 0.4, the network is unable to restore the original image.
- Instead, it converges against one of the 19 random patterns.



Discrete Hopfield Network

Possible 8 states for the three-neuron Hopfield network



Discrete Hopfield Network

- The stable state is determined by the weight matrix **W**, the current input vector **X**, and the threshold matrix **q**. If the input vector is partially incorrect or incomplete, the initial state will converge into the stable state after a few iterations.

- Suppose, for instance, that the network is required to memorize two opposite states, (1, 1, 1) and (-1, -1, -1). Thus,

$$\mathbf{Y}_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad \mathbf{Y}_2 = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} \quad \text{or} \quad \mathbf{Y}_1^T = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \quad \mathbf{Y}_2^T = \begin{bmatrix} -1 & -1 & -1 \end{bmatrix}$$

Discrete Hopfield Network

- To build the weight matrix

$$\mathbf{W} = \sum_{k=1}^{p} \mathbf{x}^k (\mathbf{x}^k)^T - p\mathbf{I} \qquad w_{ij} = \begin{cases} \sum_{k=1}^{p} x_i^k x_j^k & i \neq j \\ 0 & i = j \end{cases}$$

- Determine the weight matrix as follows:

$$\mathbf{W} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} + \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} \begin{bmatrix} -1 & -1 & -1 \end{bmatrix} - 2 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 2 \\ 2 & 0 & 2 \\ 2 & 2 & 0 \end{bmatrix}$$

- Next, the network is tested by the sequence of input vectors, X_1 and X_2, which are equal to the output (or target) vectors Y_1 and Y_2, respectively.

Discrete Hopfield Network

- Activate the Hopfield network by applying input vector X and calculate the actual output vector Y

$$\mathbf{Y}_1 = sign\left\{ \begin{bmatrix} 0 & 2 & 2 \\ 2 & 0 & 2 \\ 2 & 2 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right\} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$\mathbf{Y}_2 = sign\left\{ \begin{bmatrix} 0 & 2 & 2 \\ 2 & 0 & 2 \\ 2 & 2 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right\} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$$

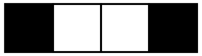
- Compare the result with the initial input vector X

Discrete Hopfield Network

- The remaining six states are all unstable. However, stable states (also called **fundamental memories**) are capable of attracting states that are close to them.
- The fundamental memory (1, 1, 1) attracts unstable states (-1, 1, 1), (1, -1, 1) and (1, 1, -1). Each of these unstable states represents a single error, compared to the fundamental memory (1, 1, 1).
- The fundamental memory (-1, -1, -1) attracts unstable states (-1, -1, 1), (-1, 1, -1) and (1, -1, -1).
- Thus, the Hopfield network can act as an **error correction network**.

Example 1: Weights Matrix

$$\mathbf{x}^1 = (1, -1, -1, 1)^T$$



$$\mathbf{x}^2 = (-1, 1, -1, 1)^T$$

$$\mathbf{x}^1(\mathbf{x}^1)^T + \mathbf{x}^2(\mathbf{x}^2)^T = \begin{bmatrix} 2 & -2 & 0 & 0 \\ -2 & 2 & 0 & 0 \\ 0 & 0 & 2 & -2 \\ 0 & 0 & -2 & 2 \end{bmatrix}$$

$$\mathbf{x}^1(\mathbf{x}^1)^T = \begin{bmatrix} 1 & -1 & -1 & 1 \\ -1 & 1 & 1 & -1 \\ -1 & 1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

$$\mathbf{W} = \sum_{k=1}^{p} \mathbf{x}^k (\mathbf{x}^k)^T - p\mathbf{I}$$

$$\mathbf{x}^2(\mathbf{x}^2)^T = \begin{bmatrix} 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \end{bmatrix}$$

$$\mathbf{W} = \begin{bmatrix} 0 & -2 & 0 & 0 \\ -2 & 0 & 0 & 0 \\ 0 & 0 & 0 & -2 \\ 0 & 0 & -2 & 0 \end{bmatrix}$$

Example 2: Spurious State

- Use a 4-node Hopfield network to store 3 patterns:

  (1 1 -1 -1), (1 1 1 1) and (-1 -1 1 1)

  - weights:

$$\begin{pmatrix} 0 & 1 & -1/3 & -1/3 \\ 1 & 0 & -1/3 & -1/3 \\ -1/3 & -1/3 & 0 & 1 \\ -1/3 & -1/3 & 1 & 0 \end{pmatrix}$$

- Corrupted input pattern: (-1 -1 -1 -1)
  - if node 4 is randomly selected:

  $(-1/3\ -1/3\ 1\ 0)\ (-1\ -1\ -1\ -1)^T + (-1) = 1/3 + 1/3 - 1 - 0 - 1 = -4/3$
  - no change of state for node 4
  - same for all other nodes, net stabilized at (-1 -1 -1 -1)
  - a spurious state is recalled

Example 2: Spurious State

- For input pattern (-1 -1 -1 0)
  - if node 4 is selected first

$(-1/3\ -1/3\ 1\ 0)\ (-1\ -1\ -1\ 0)^T + (0) = 1/3 + 1/3 - 1 - 0 - 0 = -1/3$

  - node 4 changes state to -1: (-1 -1 -1 -1)
  - network stabilizes at (-1 -1 -1 -1)
  - however, if the node selection sequence is 1>2>3>4, the net stabilizes at state (-1 -1 1 1): a correct pattern

$$W = \begin{pmatrix} 0 & 1 & -1/3 & -1/3 \\ 1 & 0 & -1/3 & -1/3 \\ -1/3 & -1/3 & 0 & 1 \\ -1/3 & -1/3 & 1 & 0 \end{pmatrix}$$

-1	-1	-1	0
-1	-1	-1	0
-1	-1	-1	0
-1	-1	1	0
-1	-1	1	1
-1	-1	1	1
-1	-1	1	1
-1	-1	1	1
-1	-1	1	1

Limitations of Hopfield Network

- The number of patterns that can be stored and accurately recalled is severely limited
  - net may converge to a novel spurious pattern

- Exemplar pattern will be unstable if it shares many bits in common with another exemplar pattern

THANK YOU

VISIT US

WWW.XJTLU.EDU.CN

FOLLOW US

@XJTLU

Xi'an Jiaotong-Liverpool University
西交利物浦大学