ASSOCIATIVE MEMORIES & HOPFIELD NETWORK

INT301 Bio-computation, Week 12, 2025

Memory in Computer System

- Standard computer memory is accessed through **assigned addresses**.
- When a user searches for a file, the CPU must convert the request to a numerical instruction and then search through the memory for the corresponding address
- A computer's memory is most commonly referred to as RAM (random access memory).

Associative Memory & Pattern Association

- An associative memory is a **content-addressable structure** that maps a set of input patterns to a set of output patterns.

- **That is, memory can be directly accessed by the content, rather than the physical address in the memory.**

Associative Memory & Pattern Association

- Associative memory is often linked to pattern association
  - Associating patterns which are
    - similar
    - contrary
    - in close proximity (spatial)
    - in close succession (temporal)
  - Associative recall
    - evoke associated patterns
    - recall a pattern by part of it
    - evoke/recall with incomplete/noisy patterns

Associative Memories

- **Two types of associative memory**: ***auto-associative*** and ***hetero-associative***.
- Auto-association
  - retrieves a previously stored pattern that most closely resembles the current pattern.
- Hetero-association
  - the retrieved pattern is, in general, different from the input pattern not only in content but possibly also in type and format.

Associative Memories

Auto-association

memory

Hetero-association

memory

Former
US President

memory

Associative Memories

Stored Patterns

$$(\mathbf{x}^1, \mathbf{y}^1)$$
$$(\mathbf{x}^2, \mathbf{y}^2)$$
$$\vdots$$
$$(\mathbf{x}^p, \mathbf{y}^p)$$

$$\mathbf{x}^i \in R^n$$

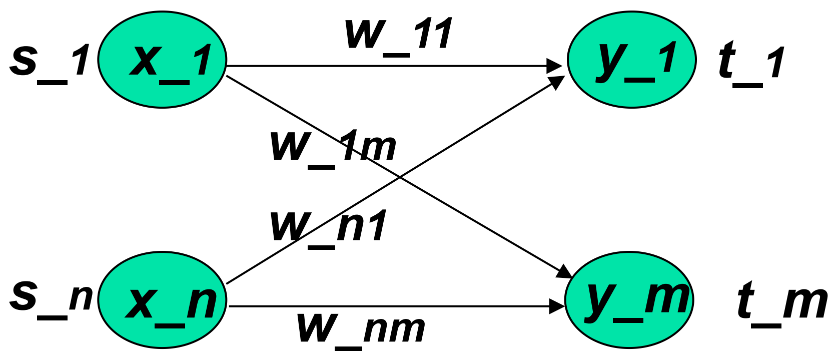
$$\mathbf{y}^i \in R^m$$

$$\mathbf{x}^i \equiv \mathbf{y}^i$$ Autoassociative

$$\mathbf{x}^i \neq \mathbf{y}^i$$ Heteroassociative

Simple AM

- Network structure: single layer
  - one output layer of non-linear units and one input layer
  - similar to the simple network for classification



- Goal of learning:
  - to obtain a set of weights ***w_ij***
  - from a set of training pattern pairs {***s*** **:** ***t***}
  - such that when ***s*** is applied to the input layer, ***t*** is computed at the output layer

Simple AM

- Similar to Hebbian learning for classification
- Algorithm: (bipolar or binary patterns)
  - For each training samples ***s* : *t*** $\Delta w_{ij} = s_i \cdot t_j$
  - Δw_{ij} increases if both input and output are ON (binary) or have the same sign (bipolar)
- If $\Delta w_{ij} = 0$ initially, then after updates for all P training patterns

$$w_{ij} = \sum_{p=1}^{P} s_i(p) t_j(p) \quad W = \{w_{ij}\}$$

- Instead of obtaining ***W*** by iterative updates, it can be computed from the training set by calculating the outer product of ***s*** and ***t***.

Simple AM

- **Outer product**: Let ***s*** and ***t*** be **row** vectors.

  Then for a particular training pair ***s:t***

$$\Delta W(p) = s^T(p) \otimes t(p) = \begin{bmatrix} s_1 \\ \\ \\ s_n \end{bmatrix} [t_1, \ldots\ldots t_m] = \begin{bmatrix} s_1 t_1 \ldots\ldots s_1 t_m \\ s_2 t_1 \ldots\ldots s_2 t_m \\ \\ s_n t_1 \ldots\ldots s_n t_m \end{bmatrix} = \begin{bmatrix} \Delta w_{11} \ldots\ldots \Delta w_{1m} \\ \\ \\ \Delta w_{n1} \ldots\ldots \Delta w_{nm} \end{bmatrix}$$

  and $W(P) = \sum_{p=1}^{P} s^T(p) \otimes t(p)$

- It involves 3 nested loops p, i, j (order of p is irrelevant)

```
p= 1 to P                  /* for every training pair */
    i = 1 to n             /* for every row in W    */
        j = 1 to m         /* for every element j in row i */
```

$$w_{ij} := w_{ij} + s_i(p) \cdot t_j(p)$$

Simple AM

- Does this method provide a good association?
  - Recall with training samples (after the weights are learned or computed)
  - Apply $\boldsymbol{s}(\boldsymbol{k})$ to one layer, hope $\boldsymbol{t}(\boldsymbol{k})$ appear on the other, e.g. $\boldsymbol{f}(\boldsymbol{s}(\boldsymbol{k})\boldsymbol{W}) = \boldsymbol{t}(\boldsymbol{k})$
  - May not always succeed (each weight contains some information from all samples)

$$s(k)W = s(k)\sum_{p=1}^{P} s^T(p)t(p) = \sum_{p=1}^{P} s(k)s^T(p)t(p)$$

$$= s(k)s^T(k)t(k) + \sum_{p \neq k} s(k)s^T(p)t(p)$$

$$= \|s(k)\|^2\, t(k) + \sum_{p \neq k} s(k)s^T(p)t(p)$$

principal term

cross-talk term

Simple AM

- Principal term gives the association between ***s*(*k*)** and ***t*(*k*)**.
- Cross-talk represents correlation between ***s*(*k*):*t*(*k*)** and other training pairs. When cross-talk is large, ***s*(*k*)** will recall something other than ***t*(*k*)**.
- If all ***s*(*p*)** are orthogonal to each other, then $s(k)s^T(p)=0$, no sample other than ***s*(*k*):*t*(*k*)** contribute to the result.
- However, there are at most ***n*** orthogonal vectors in an *n*-dimensional space.
- Cross-talk increases when ***P*** increases.

Example 1: hetero-associative

- Binary pattern pairs ***s*** **:** ***t*** with **|*s*|** = 4 and **|*t*|** = 2.
- Total weighted input to output units: $y\_in_j = \sum_i x_i w_{ij}$
- Activation function: threshold

$$y_j = \begin{cases} 1 & \text{if} \quad y\_in_j > 0 \\ 0 & \text{if} \quad y\_in_j \le 0 \end{cases}$$

- Weights are computed (sum of outer products of all training pairs)

$$W = \sum_{p=1}^{P} s_i^T(p) t_j(p)$$

- Training samples:

	s(p)	t(p)
p=1	(1 0 0 0)	(1, 0)
p=2	(1 1 0 0)	(1, 0)
p=3	(0 0 0 1)	(0, 1)
p=4	(0 0 1 1)	(0, 1)

Example 1: hetero-associative

$$s^T(1)\otimes t(1)=\begin{pmatrix}1\\0\\0\\0\end{pmatrix}\begin{pmatrix}1 & 0\end{pmatrix}=\begin{pmatrix}1 & 0\\0 & 0\\0 & 0\\0 & 0\end{pmatrix} \qquad s^T(2)\otimes t(2)=\begin{pmatrix}1\\1\\0\\0\end{pmatrix}\begin{pmatrix}1 & 0\end{pmatrix}=\begin{pmatrix}1 & 0\\1 & 0\\0 & 0\\0 & 0\end{pmatrix}$$

$$s^T(3)\otimes t(3)=\begin{pmatrix}0\\0\\0\\1\end{pmatrix}\begin{pmatrix}0 & 1\end{pmatrix}=\begin{pmatrix}0 & 0\\0 & 0\\0 & 0\\0 & 1\end{pmatrix} \qquad s^T(4)\otimes t(4)=\begin{pmatrix}0\\0\\1\\1\end{pmatrix}\begin{pmatrix}0 & 1\end{pmatrix}=\begin{pmatrix}0 & 0\\0 & 0\\0 & 1\\0 & 1\end{pmatrix}$$

Computing the weights $\quad \boldsymbol{W}=\sum_{p=1}^{P}\boldsymbol{s}_i{}^{T}(\boldsymbol{p})\boldsymbol{t}_j(\boldsymbol{p})$

$$W=\begin{pmatrix}2 & 0\\1 & 0\\0 & 1\\0 & 2\end{pmatrix}$$

Example 1: hetero-associative

x=(1 0 0 0)

$$\begin{pmatrix}1 & 0 & 0 & 0\end{pmatrix}\begin{pmatrix}2 & 0\\ 1 & 0\\ 0 & 1\\ 0 & 2\end{pmatrix} = \begin{pmatrix}2 & 0\end{pmatrix}$$

$$y_1 = 1, \quad y_2 = 0$$

x=(0 1 0 0) similar to S(1) and S(2)

$$\begin{pmatrix}0 & 1 & 0 & 0\end{pmatrix}\begin{pmatrix}2 & 0\\ 1 & 0\\ 0 & 1\\ 0 & 2\end{pmatrix} = \begin{pmatrix}1 & 0\end{pmatrix}$$

$$y_1 = 1, \quad y_2 = 0$$

x=(0 1 1 0)

$$\begin{pmatrix}0 & 1 & 1 & 0\end{pmatrix}\begin{pmatrix}2 & 0\\ 1 & 0\\ 0 & 1\\ 0 & 2\end{pmatrix} = \begin{pmatrix}1 & 1\end{pmatrix}$$

$$y_1 = 1, \quad y_2 = 1$$

(1 0 0 0), (1 1 0 0) class (1, 0)
(0 0 0 1), (0 0 1 1) class (0, 1)

(0 1 1 0) is not sufficiently similar to any class

Example 2: auto-associative

- Same as hetero-associative nets, except ***t*(*p*) = *s* (*p*)**.
- Used to recall a pattern by its noisy or incomplete version. (**pattern completion/pattern recovery**)
- A single pattern ***s*** = (1, 1, 1, -1) is stored (weights computed by outer product)

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

training pat. $\quad (1\,1\,1\,-1)\cdot\boldsymbol{W} = (4\,4\,4\,-4) \to (1\,1\,1\,-1)$

noisy pat $\quad (-1\,1\,1\,-1)\cdot\boldsymbol{W} = (2\,2\,2\,-2) \to (1\,1\,1\,-1)$

missing info $\quad (0\,0\,1\,-1)\cdot\boldsymbol{W} = (2\,2\,2\,-2) \to (1\,1\,1\,-1)$

more noisy $\quad (-1\,-1\,1\,-1)\cdot\boldsymbol{W} = (0\,0\,0\,0)$ not recognized

Example 2: auto-associative

- ***W*** is always a symmetric matrix
- Diagonal elements will dominate the computation when multiple patterns are stored (= P).
- When P is large, ***W*** is close to an identity matrix. This causes output = input, which may not be any stoned pattern. The pattern correction power is lost.
- Replace diagonal elements by zero:

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

$$(1 \quad 1 \quad 1 \quad -1)W' = (3 \quad 3 \quad 3 \quad -3) \to (1 \quad 1 \quad 1 \quad -1)$$

$$(-1 \quad 1 \quad 1 \quad -1)W' = (3 \quad 1 \quad 1 \quad -1) \to (1 \quad 1 \quad 1 \quad -1)$$

$$(0 \quad 0 \quad 1 \quad -1)W' = (2 \quad 2 \quad 1 \quad -1) \to (1 \quad 1 \quad 1 \quad -1)$$

$$(-1 \quad -1 \quad 1 \quad -1)W' = (1 \quad 1 \quad -1 \quad 1) \to \quad wrong$$

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.

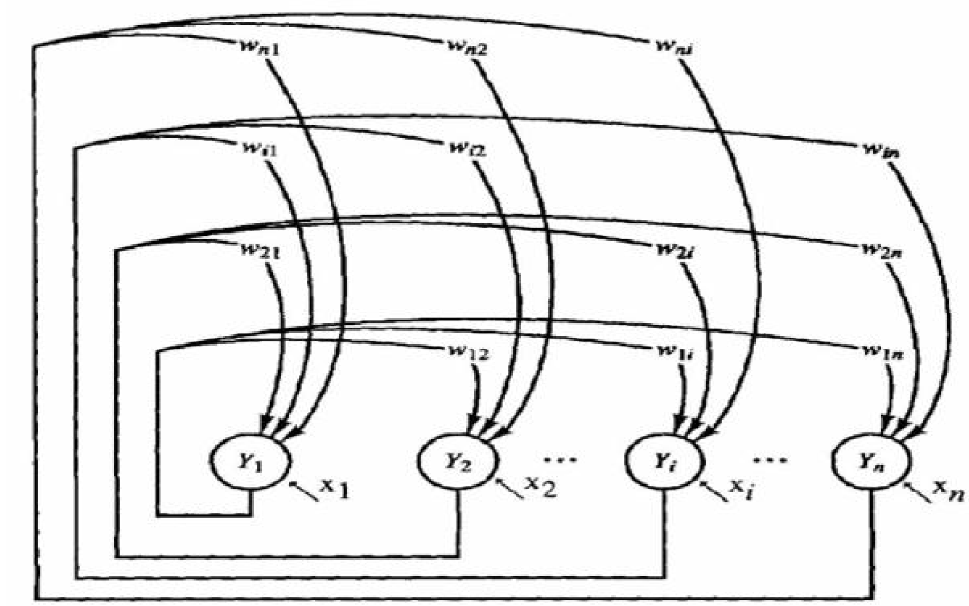
Discrete Hopfield Network

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

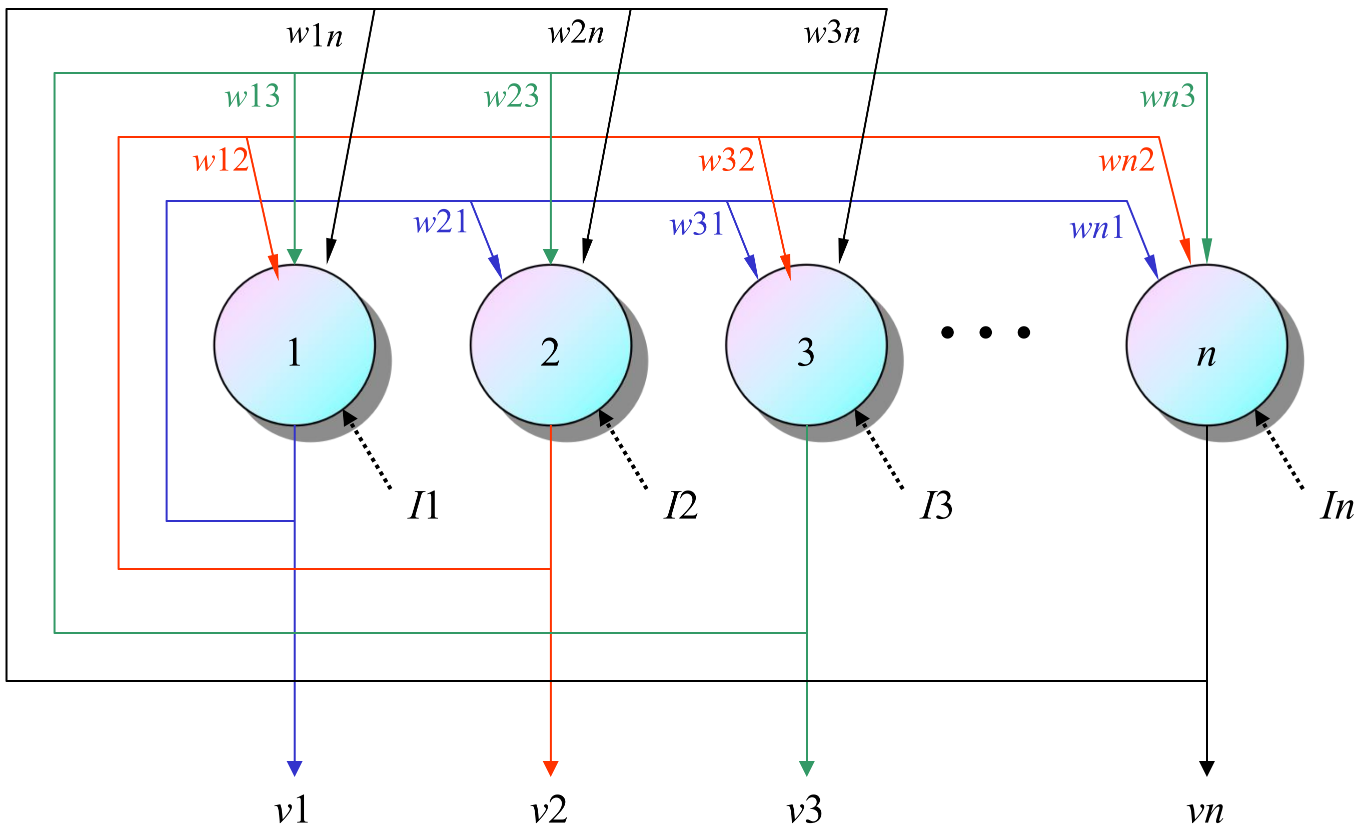
$$w_{ij} = w_{ji}$$

$$w_{ii} = 0$$

  - 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:

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

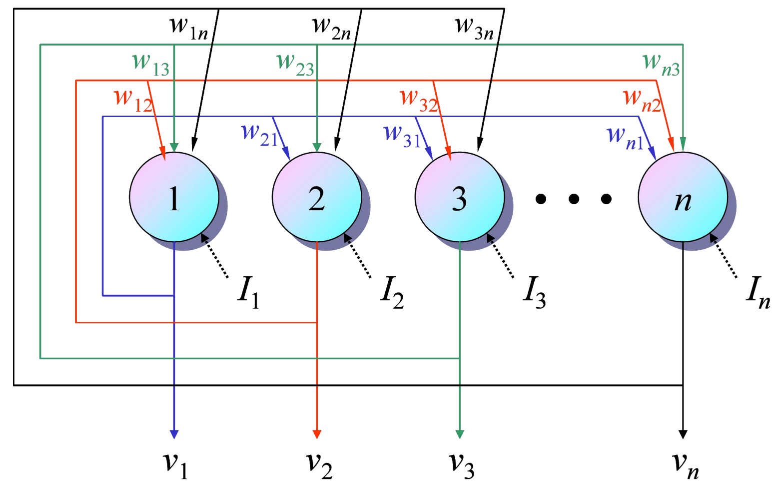
- This recursion can be performed over and over again.

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?

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

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

- **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

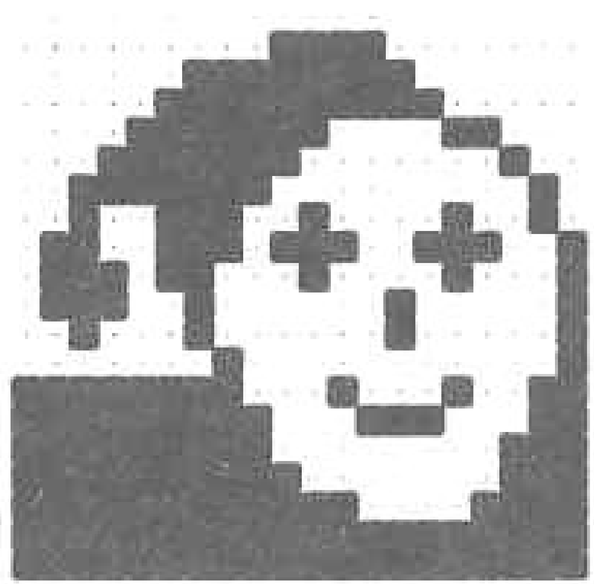
  This is because the AM has only 2 attractors

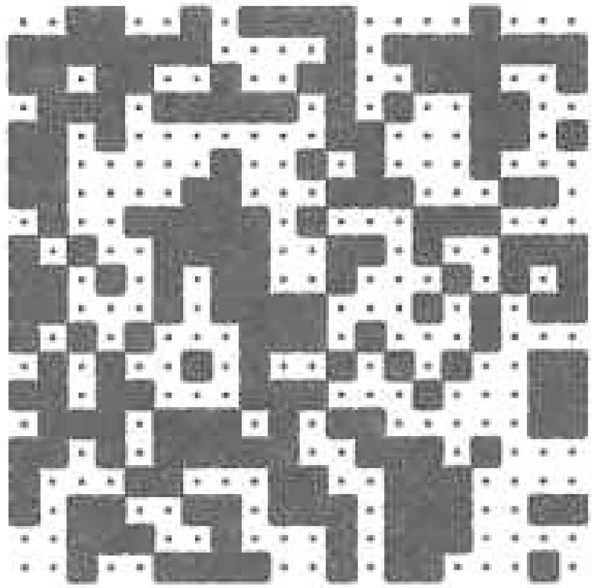
  (1 1 1 1) and (-1 -1 -1 -1)

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

Example

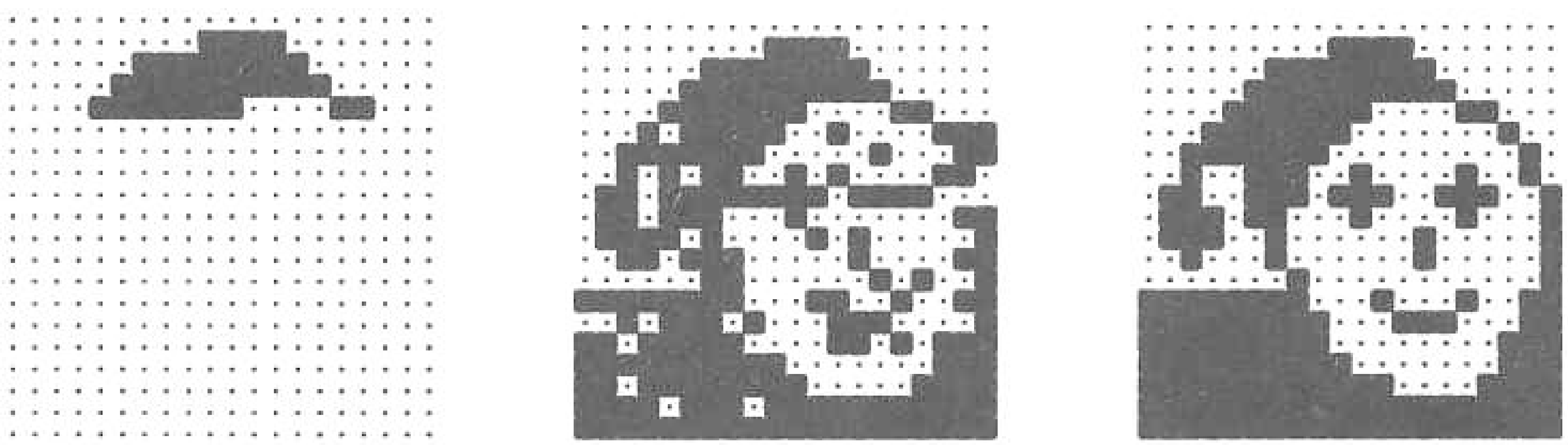
- Image reconstruction (Ritter, Schulten, Martinetz 1990)
- A 20×20 discrete Hopfield network was trained with 20 input patterns, including the one shown in the left figure and 19 random patterns as the one on the right.





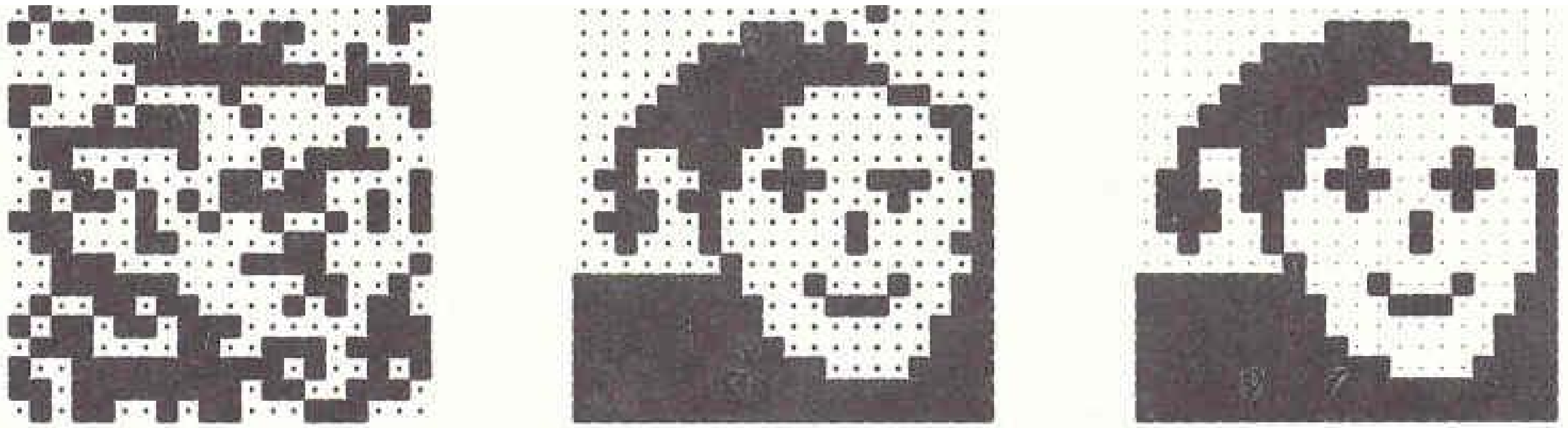
Example

- 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.



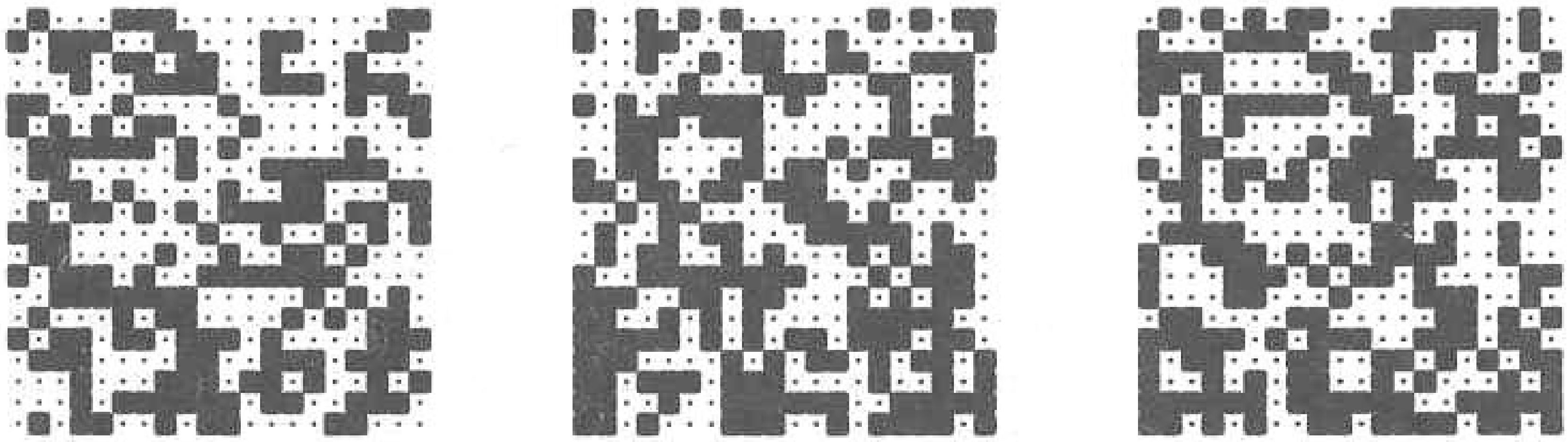
Example

- 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.



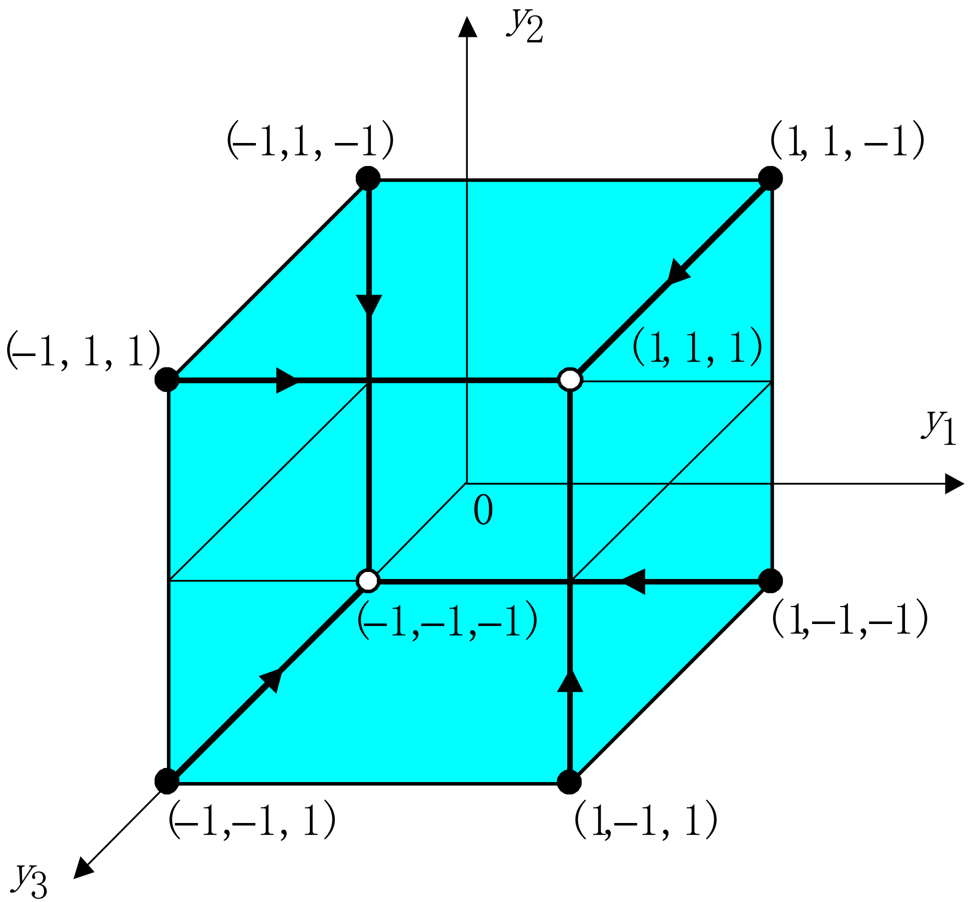
Example

- 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,

$$Y_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad Y_2 = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} \quad \text{or} \quad Y_1^T = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \quad Y_2^T = \begin{bmatrix} -1 & -1 & -1 \end{bmatrix}$$

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

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

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

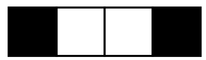
- Compare the result with the initial input vector X

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

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



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

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

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

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

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

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

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