دانشگاه کردستان
University of Kurdistan
زانکۆی کوردستان

Department of Computer Engineering
University of Kurdistan

Neural Networks (Graduate level)

Associative Networks

By: Dr. Alireza Abdollahpouri

Associative networks

- To an extent, learning is forming associations.
- Human memory associates
  - similar items,
  - contrary/opposite items,
  - items close in proximity,
  - items close in succession (e.g., in a song)

Associative networks

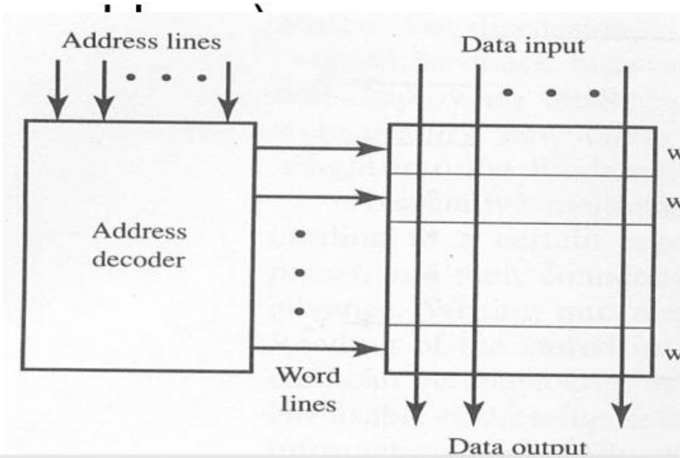
- The patterns we associate together may be
  – of the **same type** or sensory modality (e.g. a visual image may be associated with another visual image)
  – or of **different types** (e.g. a fragrance may be associated with a visual image or a feeling).
- Memorization of a pattern (or a group of patterns) may be considered to be associating the pattern with itself.

Associative networks

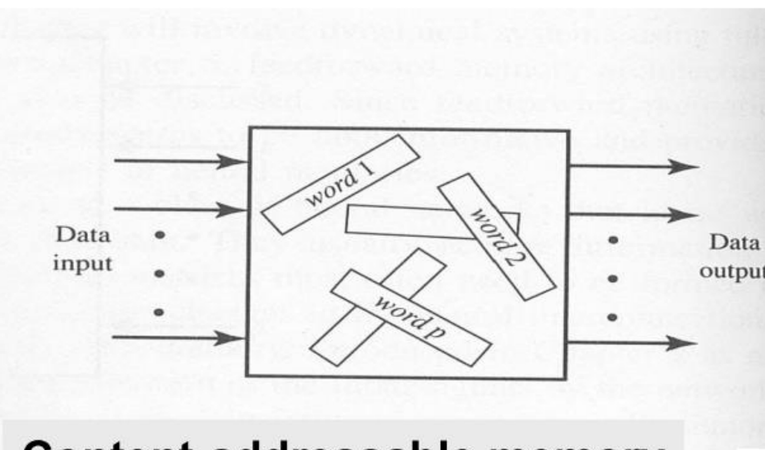
An associative network is a single-layer Network in which the weights are determined in such a way that the net can store a set of pattern associations.

Associative networks

- An associative memory (AM) net may serve as a highly simplified model of human memory. However, we shall not address the question whether they are all realistic models.
- AM provide an approach of storing and retrieving data based on content rather than storage address (info. storage in a NN is distributed throughout the system in the net's weights, hence a pattern does not have a storage address.)

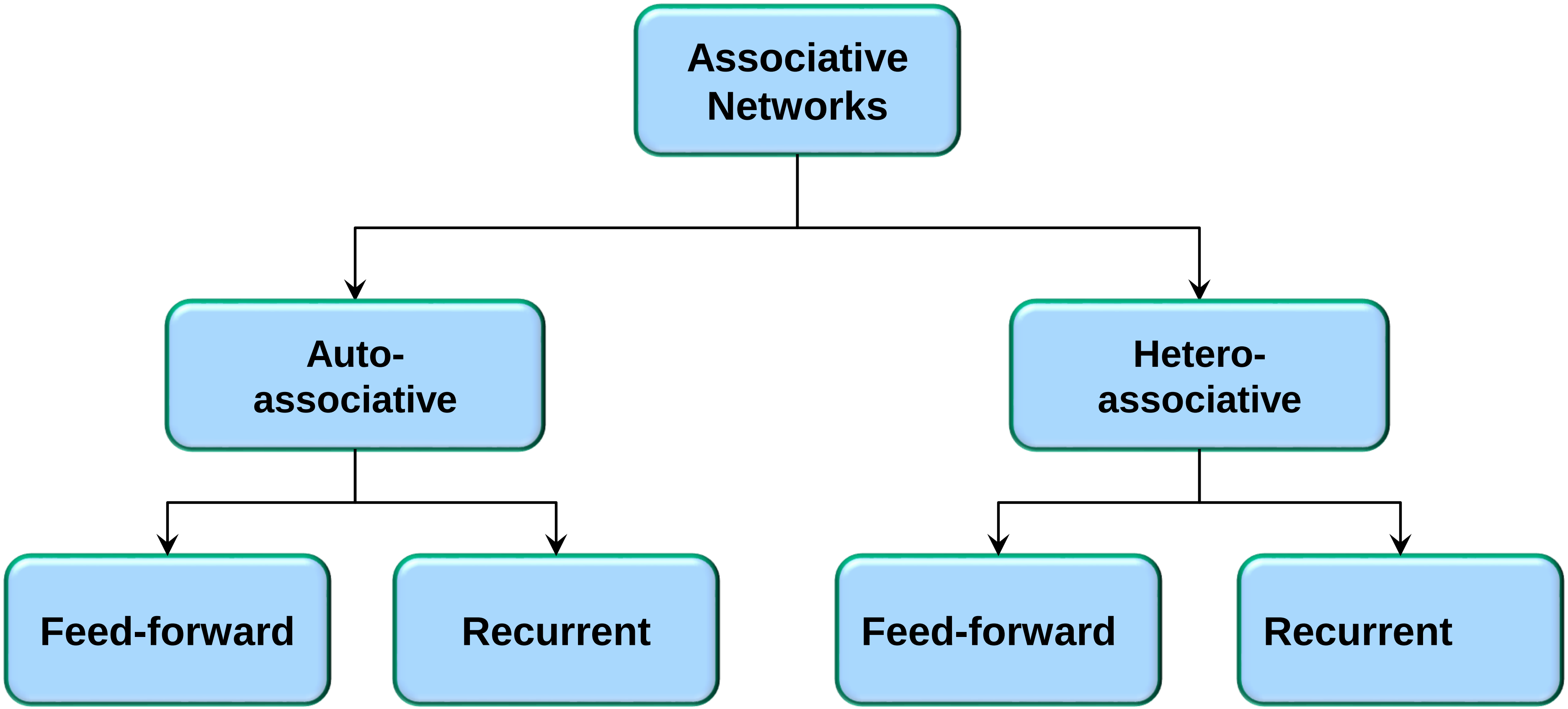


Address-addressable memory

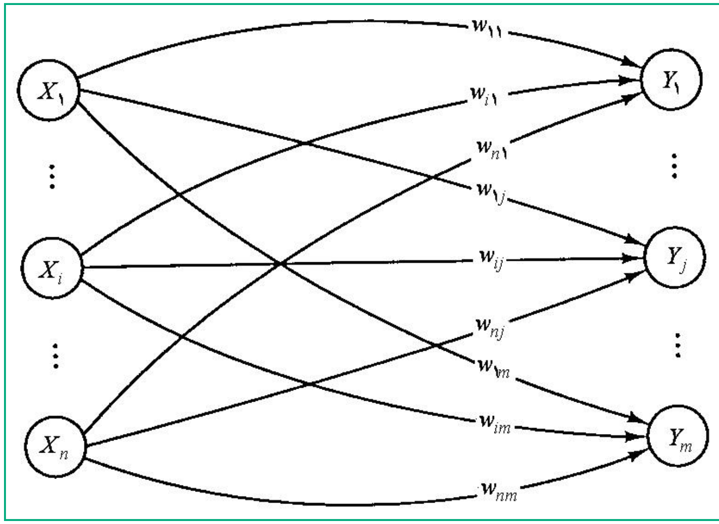


Content-addressable memory

Types of Associative networks

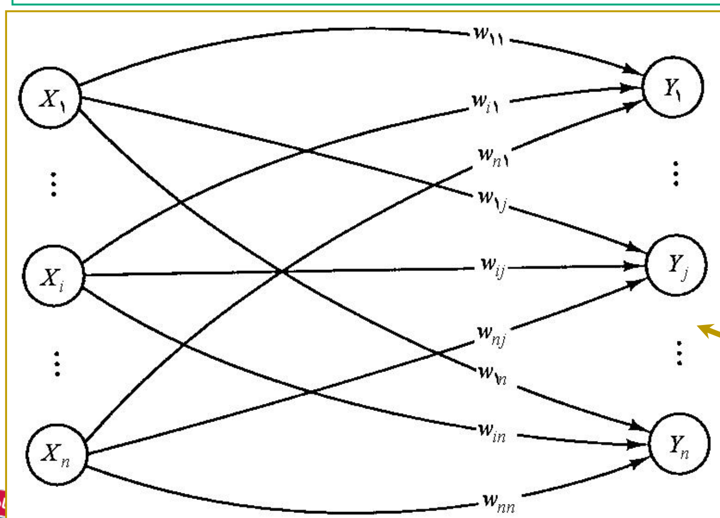


Types of Associative networks



- Hetero associative

n Input m Output

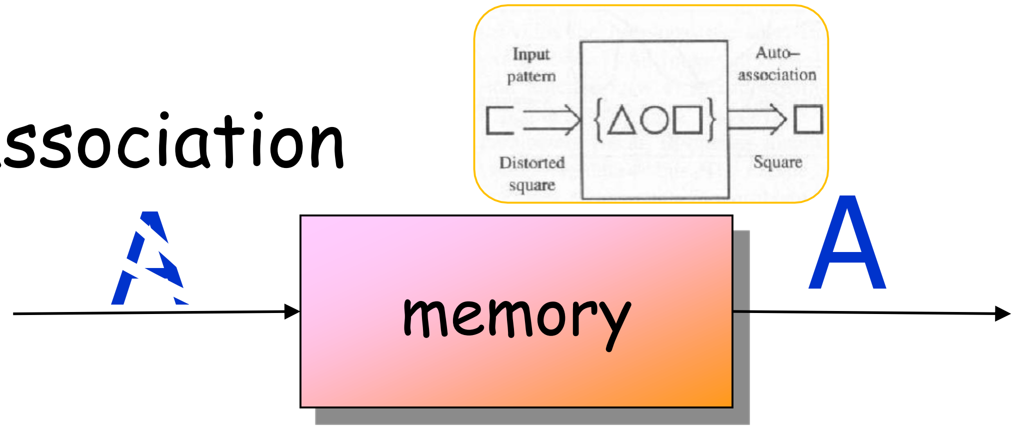


- Auto associative

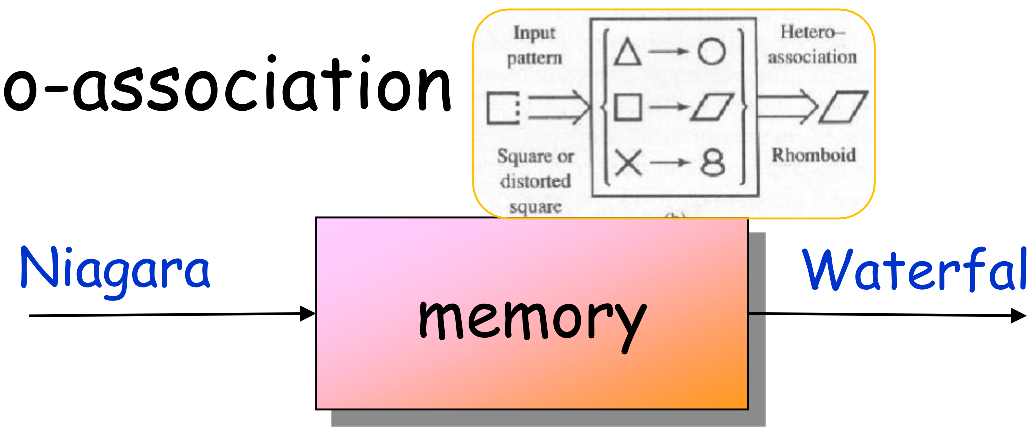
n Input n Output

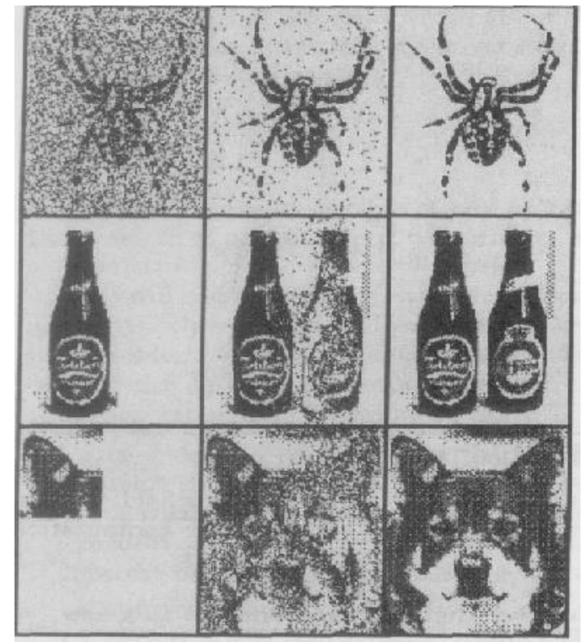
Types of Associative networks

Auto-association



Hetero-association





Associative networks

- Whether auto- or hetero-associative, the net can associate not only the exact pattern pairs used in training, but is also able to obtain associations if the input is similar to one on which it has been trained.

- Information recording: A large set of patterns (the priori information) are stored (memorized)
- Information retrieval/recall: Stored prototypes are excited according to the input key patterns

AM- Representation

- Before training an AM NN, the original patterns must be converted to an appropriate representation for computation
- In a simple example, the original pattern might consist of “on” and “off” signals, and the conversion could be:

  “on” → +1 , “off” → 0 (binary representation) or

  “on” → +1, “off” → -1 (bipolar representation).

Examples of Hetero-association

Mapping from 4-inputs to 2-outputs.

Whenever the net is shown a 4-bit input pattern, it produces a 2-bit output pattern

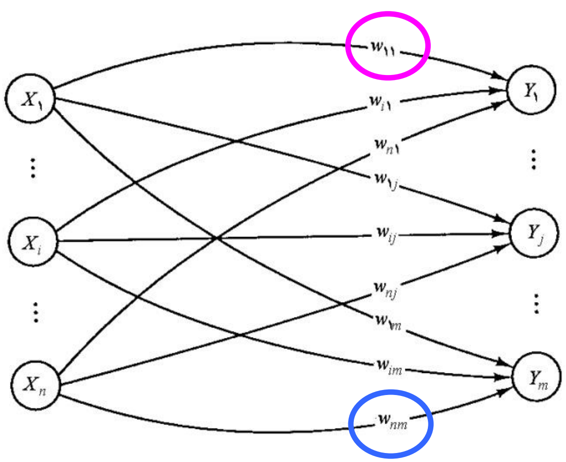
Input	Output
1 0 0 0	1 0
1 1 0 0	1 0
0 0 0 1	0 1
0 0 1 1	0 1

Hebbian Learning

Input n dimensional

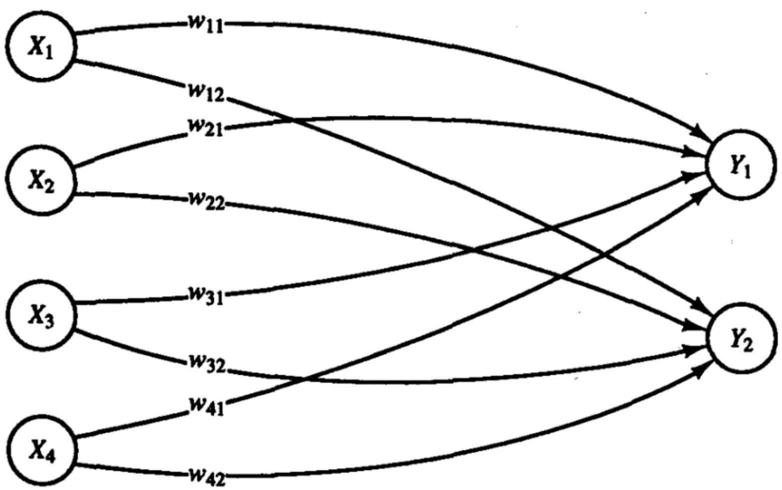
Output: m dimensional

$$S \times T = \begin{bmatrix} s_1 \\ \vdots \\ s_i \\ \vdots \\ s_n \end{bmatrix} [t_1 \ldots t_j \ldots t_m] = \begin{bmatrix} s_1 t_1 & \cdots & s_1 t_j & \cdots & s_1 t_m \\ \vdots & . & \vdots & . & \vdots \\ s_i t_1 & \cdots & s_i t_j & \cdots & s_i t_m \\ \vdots & . & \vdots & . & \vdots \\ s_n t_1 & \cdots & s_n t_j & \cdots & s_n t_m \end{bmatrix}$$



Weight matrix- using Hebb rule

Examples of Hetero-association



		s_1	s_2	s_3	s_4		t_1	t_2
1	**s**	(1,	0,	0,	0)	**t**	(1,	0)
2	**s**	(1,	1,	0,	0)	**t**	(1,	0)
3	**s**	(0,	0,	0,	1)	**t**	(0,	1)
4	**s**	(0,	0,	1,	1)	**t**	(0,	1)

Examples of Hetero-association

weight matrix to store the first pair:

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

$$\mathbf{s}:\mathbf{t}\Rightarrow(1,0,0,0):(1,0)$$

weight matrix to store the second pair:

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

$$\mathbf{s}:\mathbf{t}\Rightarrow(1,1,0,0):(1,0)$$

Examples of Hetero-association

weight matrix to store the third pair:

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

$$\mathbf{s} : \mathbf{t} \Rightarrow (0, 0, 0, 1) : (0, 1)$$

weight matrix to store the fourth pair:

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

$$\mathbf{s} : \mathbf{t} \Rightarrow (0, 0, 1, 1) : (0, 1)$$

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

Test the network

Using the activation function: $f(x) = \begin{cases} 1 & \text{if } x > 0; \\ 0 & \text{if } x \le 0. \end{cases}$

For the first input pattern $\mathrm{x} = (1, 0, 0, 0)$

$$\begin{aligned} y\_in_1 &= x_1 w_{11} + x_2 w_{21} + x_3 w_{31} + x_4 w_{41} \\ &= 1(2) + 0(1) + 0(0) + 0(0) \\ &= 2; \\ y\_in_2 &= x_1 w_{12} + x_2 w_{22} + x_3 w_{32} + x_4 w_{42} \\ &= 1(0) + 0(0) + 0(1) + 0(2) \\ &= 0. \end{aligned}$$

$$\begin{aligned} y_1 &= f(y\_in_1) = \mathrm{f}(2) = 1; \\ y_2 &= f(y\_in_2) = f(0) = 0. \end{aligned}$$

This is the correct response for the first training pattern

Test the network

Similarly, applying the same algorithm, with x equal to each of the other three training input vectors, yields:

$$(1, 1, 0, 0) \cdot \mathbf{W} = (3, 0) \rightarrow (1, 0),$$

$$(0, 0, 0, 1) \cdot \mathbf{W} = (0, 2) \rightarrow (0, 1),$$

$$(0, 0, 1, 1) \cdot \mathbf{W} = (0, 3) \rightarrow (0, 1).$$

Test the network

Testing a hetero-associative net with input **similar** to the training input

$$(0,\ 1,\ 0,\ 0)\cdot\mathbf{W} = (1,\ 0) \rightarrow (1,\ 0).$$

Thus, the net also associates a known output pattern with this input

Testing a heteroassociative net with input that is **not similar** to the training input

$$(0\ 1,\ 1,\ 0)\cdot\mathbf{W} = (1,\ 1) \rightarrow (1,\ 1).$$

The output is not one of the outputs with which the net was trained; in other words, the net does not recognize the pattern.

Bi-polar representation

$$\begin{aligned}
\mathbf{s}(1) &= (\ 1, & -1, & -1, & -1), \qquad & \mathbf{t}(1) &= (\ 1 & -1) \\
\mathbf{s}(2) &= (\ 1, & 1, & -1, & -1), \qquad & \mathbf{t}(2) &= (\ 1, & -1) \\
\mathbf{s}(3) &= (-1, & -1, & -1, & 1), \qquad & \mathbf{t}(3) &= (-1, & 1) \\
\mathbf{s}(4) &= (-1, & -1, & 1, & 1), \qquad & \mathbf{t}(4) &= (-1, & 1)
\end{aligned}$$

$$w_{ij} = \sum_p s_i(p) t_j(p)$$

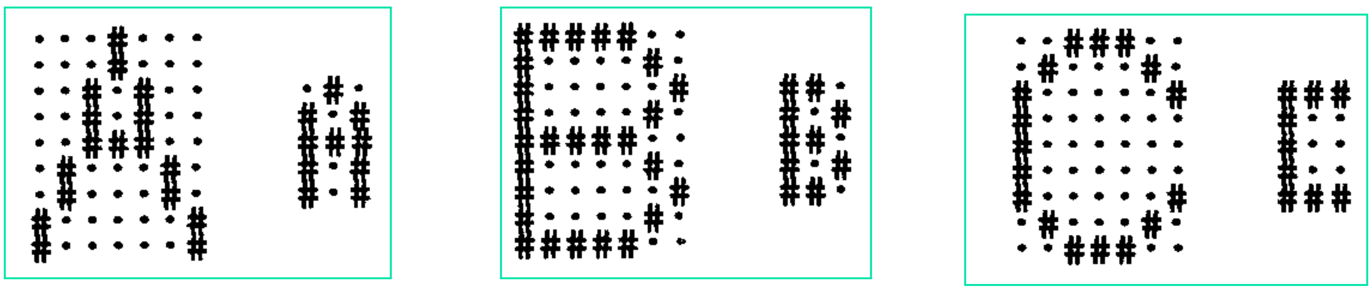
Weight matrix

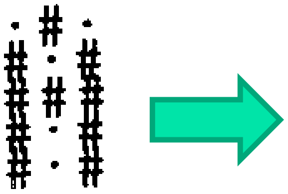
$$\left(\begin{bmatrix}1\\-1\\-1\\-1\end{bmatrix}.\begin{bmatrix}1 & -1\end{bmatrix} = \begin{bmatrix}1 & -1\\-1 & 1\\-1 & 1\\-1 & 1\end{bmatrix}\right) + \left(\begin{bmatrix}1\\1\\-1\\-1\end{bmatrix}.\begin{bmatrix}1 & -1\end{bmatrix} = \begin{bmatrix}1 & -1\\1 & -1\\-1 & 1\\-1 & 1\end{bmatrix}\right) + \left(\begin{bmatrix}-1\\-1\\-1\\1\end{bmatrix}.\begin{bmatrix}-1 & 1\end{bmatrix} = \begin{bmatrix}1 & -1\\1 & -1\\1 & -1\\-1 & 1\end{bmatrix}\right) + \left(\begin{bmatrix}-1\\-1\\1\\1\end{bmatrix}.\begin{bmatrix}-1 & 1\end{bmatrix} = \begin{bmatrix}1 & -1\\1 & -1\\-1 & 1\\-1 & 1\end{bmatrix}\right) = \begin{bmatrix}4 & -4\\2 & -2\\-2 & 2\\-4 & 4\end{bmatrix}$$

First input Second input Third input Fourth input

Hetero-associative network- example

A heteroassociative net for associating letters from different fonts



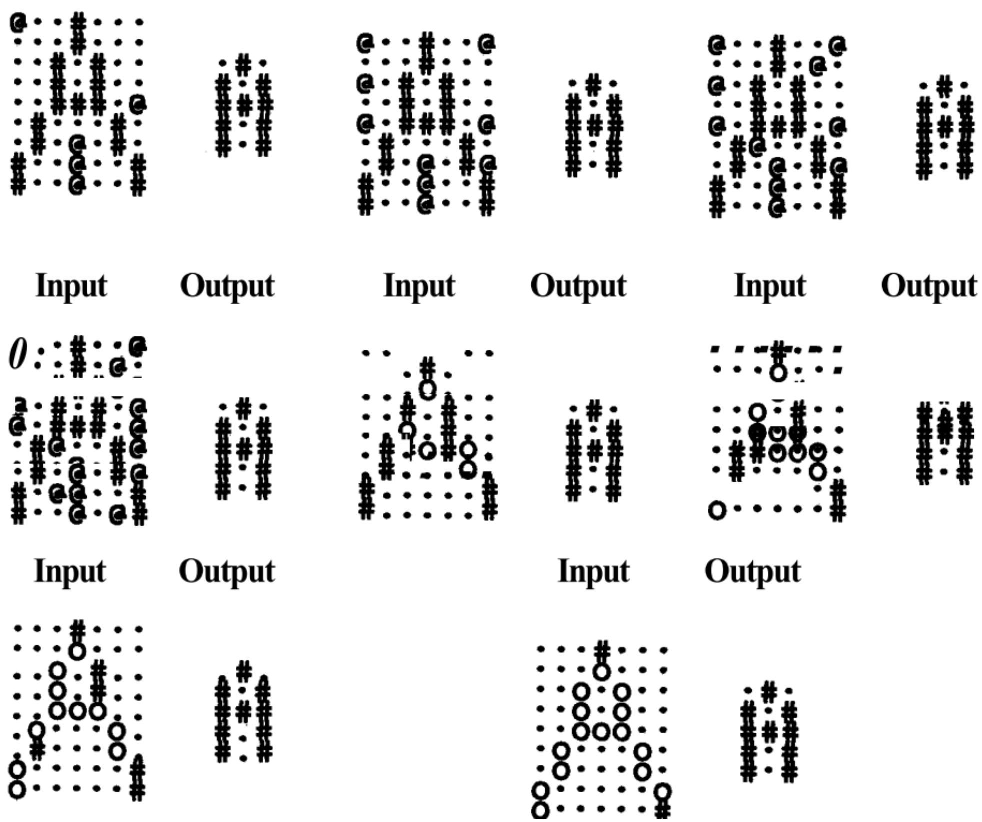


$$(-1,\ 1,\ -1,\quad 1,\ -1, 1,\quad 1, 1, 1,\quad 1,\ -1, 1,\quad 1,\ -1, 1)$$

Hetero-associative network- example

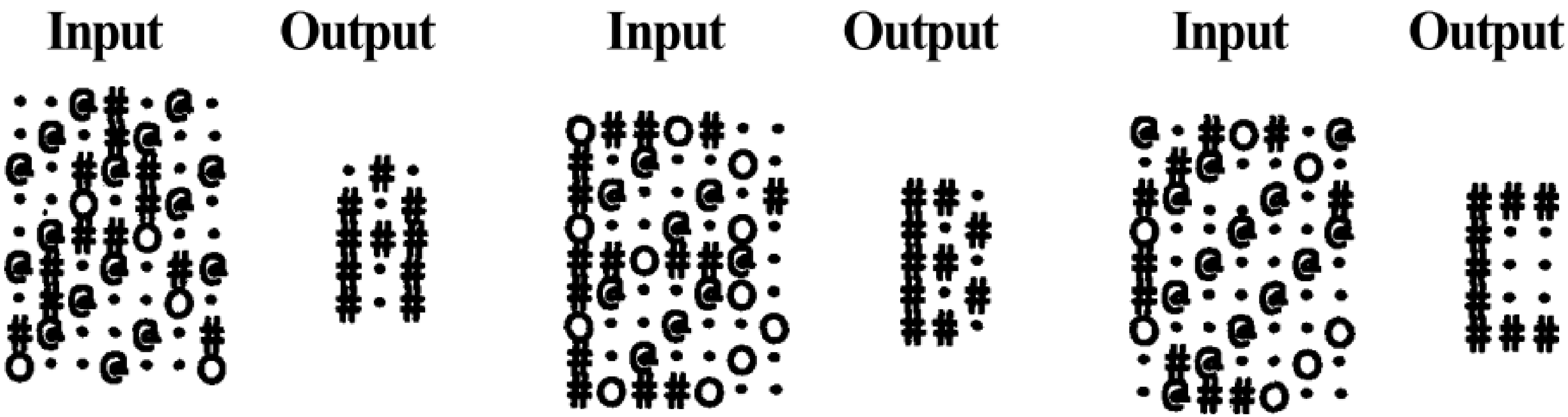
@ Pixel is now "on," but this is a mistake (noise).

O Pixel is now "off," but this is a mistake (noise).



Response of heteroassociative net to several noisy versions of pattern A.

Hetero-associative network- example



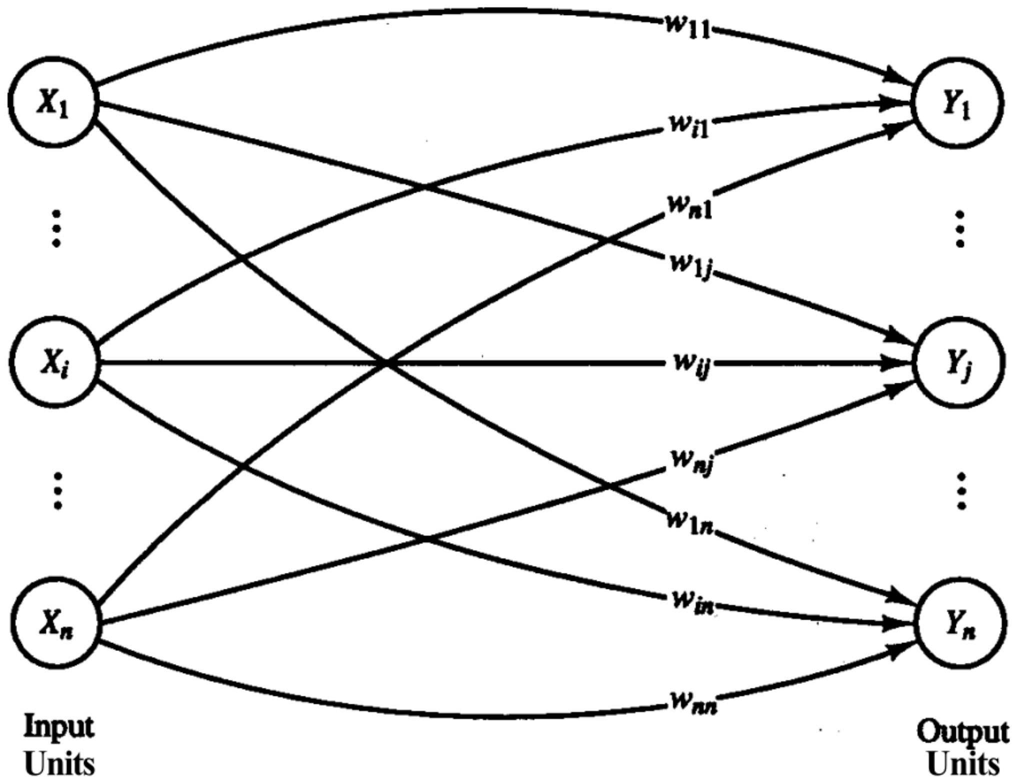
Response of heteroassociative net to patterns A, B, and C with mistakes in 1/3 of the components.

University of Kurdistan

Auto-associative nets

the weights are usually set from the formula

$$\mathbf{W} = \sum_{p=1}^{P} \mathbf{s}^{\mathrm{T}}(p)\mathbf{s}(p),$$



Auto-associative net- example

The vector s = (1, 1, 1, - 1) is stored with the weight matrix: $\mathbf{W} = \begin{bmatrix} 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & -1 \\ -1 & -1 & -1 & 1 \end{bmatrix}$

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

$$\mathbf{y}\_in = (4, 4, 4, -4)$$

$$\mathbf{y} = f(4, 4, 4, -4) = (1, 1, 1, -1)$$

$$(1,\ 1,\ 1,\ -1).\mathbf{W} = (4,\ 4,\ 4,\ -4) \rightarrow (1,\ 1,\ 1,\ -1)$$

Correct recognition of input vector

Auto-associative net- example

Testing an autoassociative net: one mistake in the input vector

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

$$\mathbf{s} = (1, 1, 1, -1)$$

$$\begin{array}{l} (-1 \quad 1, \quad 1, \quad -1) \;.\; \mathbf{W} = (2, \; 2, \; 2, \; -2) \rightarrow (1, \; 1, \; 1, \; -1) \\ (\;1 \quad -1, \quad 1, \quad -1) \;.\; \mathbf{W} = (2, \; 2, \; 2, \; -2) \rightarrow (1, \; 1, \; 1, \; -1) \\ (\;1 \quad 1, \quad -1, \quad -1) \;.\; \mathbf{W} = (2, \; 2, \; 2, \; -2) \rightarrow (1, \; 1, \; 1, \; -1) \\ (\;1 \quad 1, \quad 1, \quad 1) \;.\; \mathbf{W} = (2, \; 2, \; 2, \; -2) \rightarrow (1, \; 1, \; 1, \; -1) \end{array}$$

Correct recognition

One mistake in input vector

Auto-associative net- example

Testing an autoassociative net: two "missing" entries in the input vector

$$\mathbf{s} = (1, 1, 1, -1)$$

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

$$\begin{array}{l}
(0, \; 0, \; 1, \; -1) \cdot \mathbf{W} = (2, \; 2, \; 2, \; -2) \rightarrow (1, \; 1, \; 1, \; -1) \\
(0, \; 1, \; 0, \; -1) \cdot \mathbf{W} = (2, \; 2, \; 2, \; -2) \rightarrow (1, \; 1, \; 1, \; -1) \\
(0, \; 1, \; 1, \; 0) \cdot \mathbf{W} = (2, \; 2, \; 2, \; -2) \rightarrow (1, \; 1, \; 1, \; -1) \\
(1, \; 0, \; 0, \; -1) \cdot \mathbf{W} = (2, \; 2, \; 2, \; -2) \rightarrow (1, \; 1, \; 1, \; -1) \\
(1, \; 0, \; 1, \; 0) \cdot \mathbf{W} = (2, \; 2, \; 2, \; -2) \rightarrow (1, \; 1, \; 1, \; -1) \\
(1, \; 1, \; 0, \; 0) \cdot \mathbf{W} = (2, \; 2, \; 2, \; -2) \rightarrow (1, \; 1, \; 1, \; -1)
\end{array}$$

Correct recognition

two "missing" entries

Auto-associative net- example

Testing an autoassociative net: two mistakes in the input vector

$$\mathbf{s} = (1, 1, 1, -1) \qquad \mathbf{W} = \begin{bmatrix} 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & -1 \\ -1 & -1 & -1 & 1 \end{bmatrix}$$

$$(-1,\ -1,\ 1,\ -1).\mathbf{W} = (0,\ 0,\ 0,\ 0)$$

Incorrect recognition

Storage Capacity

- An autoassociative net with four nodes can store three orthogonal vectors

Storage Capacity

$$
\begin{array}{ccc}
(1, 1, -1, -1) & (-1, 1, 1, -1) & \\
\Downarrow & \Downarrow & \\
\mathbf{W}_1 & \mathbf{W}_2 & \mathbf{W}_1 + \mathbf{W}_2
\end{array}
$$

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

It is fairly common for an autoassociative network to have its diagonal terms set to zero

Storage Capacity

Storing two **Orthogonal** vectors in an autoassociative net

$$(1, 1, -1, -1) \qquad (-1, 1, 1, -1)$$

$$\Downarrow \qquad \Downarrow$$

$$\mathbf{W}_1 \qquad \mathbf{W}_2 \qquad \mathbf{W}_1 + \mathbf{W}_2$$

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

$$(1, 1, -1, -1).[\mathbf{W}_1 + \mathbf{W}_2] = (1, 1, -1, -1)$$

$$(-1, 1, 1, -1).[\mathbf{W}_1 + \mathbf{W}_2] = (-1, 1, 1, -1)$$

Correct recognition

Storage Capacity

Attempting to store two **non-orthogonal** vectors in an autoassociative net

$$(1, 1, -1, 1) \quad (1, -1, -1, 1)$$

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

✓ $(1, -1, -1, 1).\mathbf{W} = (1, -1, -1, 1)$

✗ $(1, 1, -1, 1).\mathbf{W} = (1, -1, -1, 1)$ ← Incorrect Recognition

Storage Capacity

Storing three **Orthogonal** vectors in an autoassociative net

$$
\begin{array}{cccc}
(1, 1, -1, -1) & (-1, 1, 1, -1) & (-1, 1, -1, 1) & \\
\Downarrow & \Downarrow & \Downarrow & \\
\mathbf{W}_1 & \mathbf{W}_2 & \mathbf{W}_3 & \mathbf{W}_1 + \mathbf{W}_2 + \mathbf{W}_3
\end{array}
$$

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

Correct recognition of all vectors

Storage Capacity

Storing four **Orthogonal** vectors in an autoassociative net

$$
\begin{matrix}
(1, 1, -1, -1) & (-1, 1, 1, -1) & (-1, 1, -1, 1) & (1, 1, 1, 1) & \\
\Downarrow & \Downarrow & \Downarrow & \Downarrow & \\
\mathbf{W}_1 & \mathbf{W}_2 & \mathbf{W}_3 & \mathbf{W}_4 & \mathbf{W}_1 + \mathbf{W}_2 + \mathbf{W}_3 + \mathbf{W}_4
\end{matrix}
$$

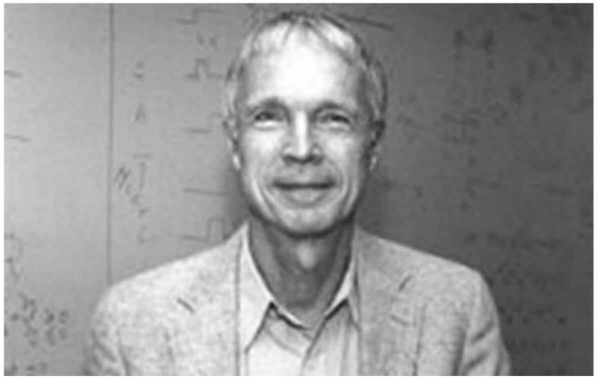
$$
\begin{bmatrix} 0 & 1 & -1 & -1 \\ 1 & 0 & -1 & -1 \\ -1 & -1 & 0 & 1 \\ -1 & -1 & 1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & -1 & -1 & 1 \\ -1 & 0 & 1 & -1 \\ -1 & 1 & 0 & -1 \\ 1 & -1 & -1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & -1 & 1 & -1 \\ -1 & 0 & -1 & 1 \\ 1 & -1 & 0 & -1 \\ -1 & 1 & -1 & 0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

learning is erased.

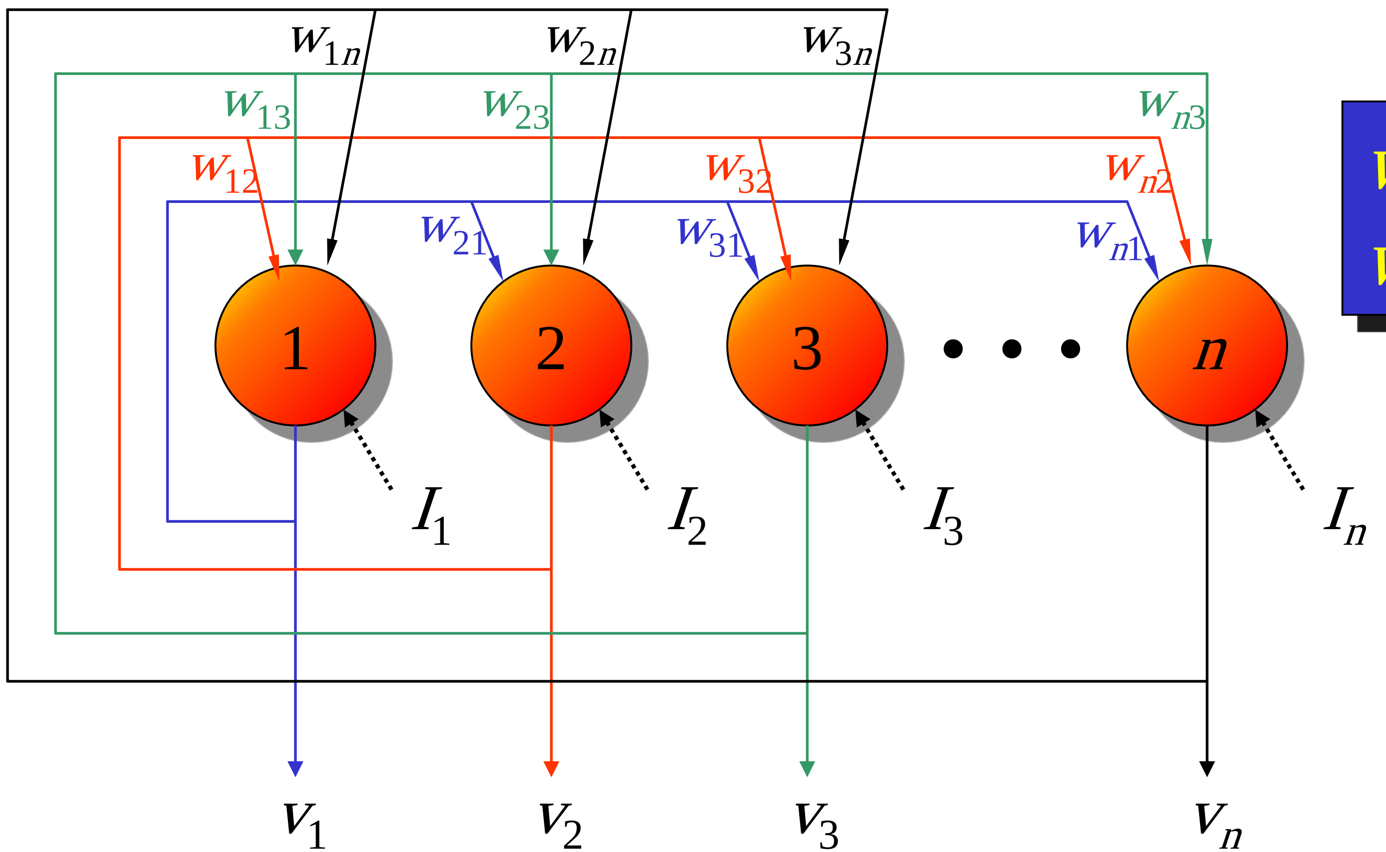
Hopfield Neural network

- Hopfield neural network (HNN) is a model of autoassociative memory
- It is a single layer neural network with feedbacks.

Output is 1 iff $\sum_j w_{ij} S_j \geq \theta_i$ and is -1 otherwise



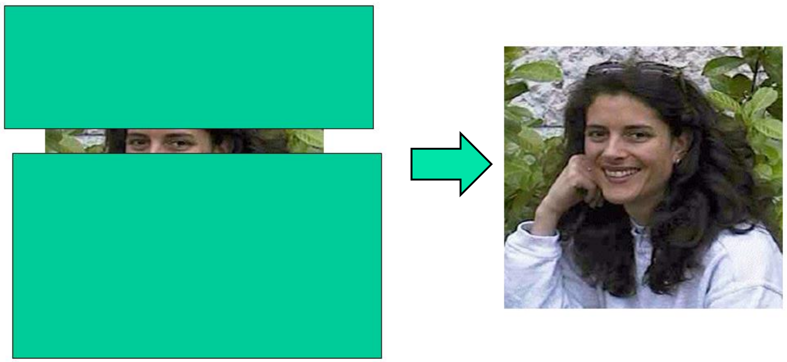
Hopfield Neural network

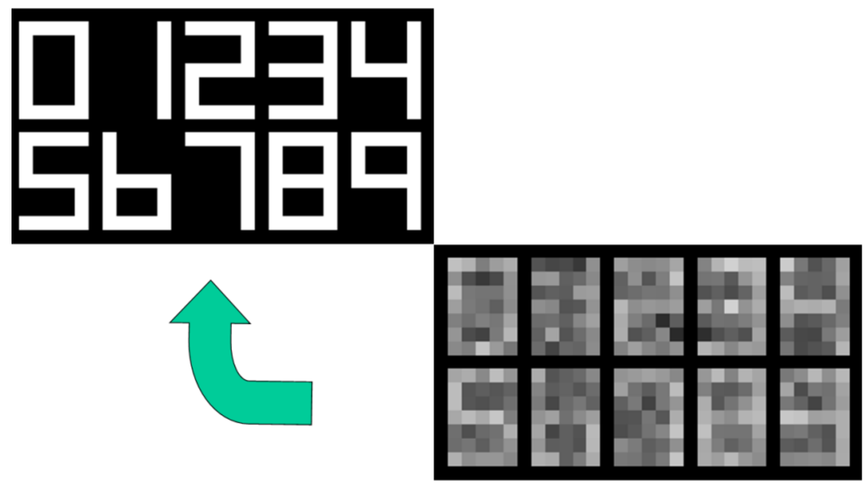


$w_{ij} = w_{ji}$

$w_{ii} = 0$

Hopfield Neural network





Hopfield Neural network

To store a set of binary patterns, the weight matrix W = is given by:

$$w_{ij} = \sum_{p} \left(2s_i(p) - 1\right)\left(2s_i(p) - 1\right), \quad i \neq j \quad ; \qquad w_{ii} = 0$$

To store a set of bipolar patterns, the weight matrix W = is given by:

$$w_{ij} = \sum_{p} s_i(p) s_j(p), \quad i \neq j \quad ; \quad w_{ii} = 0$$

Hopfield Neural network

Step 0. Initialize weights to store patterns.

While activations of the net are not converged, do Steps 1-7.

Step 1. For each input vector x, do Steps 2-6.

Step 2. Set initial activations of net equal to the external input vector x: $y_i = x_i \ , \ i = 1, \ldots, n$

Step 3. Do Steps 4-6 for each unit (Units should be updated in random order.)

Step 4. Compute net input: $y\_in_i = x_i + \sum_j y_j w_{ji}$

Step 5. Determine activation (output signal): $$y_i = \begin{cases} 1 & if \quad y\_in_i > \theta_i \\ y_i & if \quad y\_in_i = \theta_i \\ 0 & if \quad y\_in_i < \theta_i . \end{cases}$$

Step 6. Broadcast the value of y_i to all other units. (This updates the activation vector.)

Step 7. Test for convergence.

Hopfield Neural network - example

The vector (1, 1, 1,0) (or its bipolar equivalent (1, 1, 1, - 1)) was stored in a net

the weight matrix is bipolar

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

The input vector is x = (0, 0, 1, 0)

For this example the update order is $Y_1 \quad Y_4 \quad Y_3 \quad Y_2$

Hopfield Neural network - example

$y = (0, 0, 1, 0)$

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

Choose unit Y_1 to update its activation:

$$y\_in_1 = x_1 + \sum_j y_j w_{j1} = 0 + 1$$

$$y\_in_1 > 0 \;\rightarrow\; y_1 = 1 \quad ⇨ \quad y = (1, 0, 1, 0)$$

Choose unit Y_4 to update its activation:

$$y\_in_4 = x_4 + \sum_j y_j w_{j4} = 0 + (-2)$$

$$y\_in_4 < 0 \;\rightarrow\; y_4 = 0 \quad ⇨ \quad y = (1, 0, 1, 0)$$

Hopfield Neural network - example

Choose unit Y_3 to update its activation: $y = (1, 0, 1, 0)$

$$y\_in_3 = x_3 + \sum_j y_j w_{j3} = 1 + 1$$

$$y\_in_3 > 0 \rightarrow y_3 = 1 \Rightarrow y = (1, 0, 1, 0)$$

Choose unit Y_2 to update its activation:

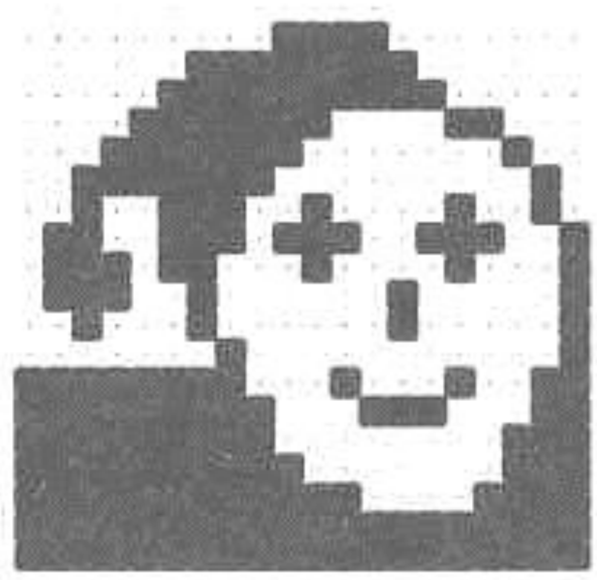
$$y\_in_2 = x_2 + \sum_j y_j w_{j2} = 0 + 2$$

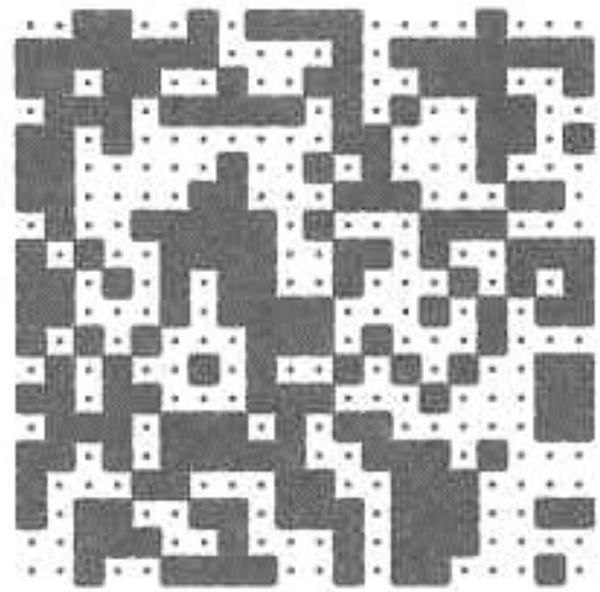
$$y\_in_2 > 0 \rightarrow y_2 = 1 \Rightarrow y = (1, 1, 1, 0)$$

further iterations do not change the activation of any unit. The net has converged to the stored vector.

Hopfield Neural network - example

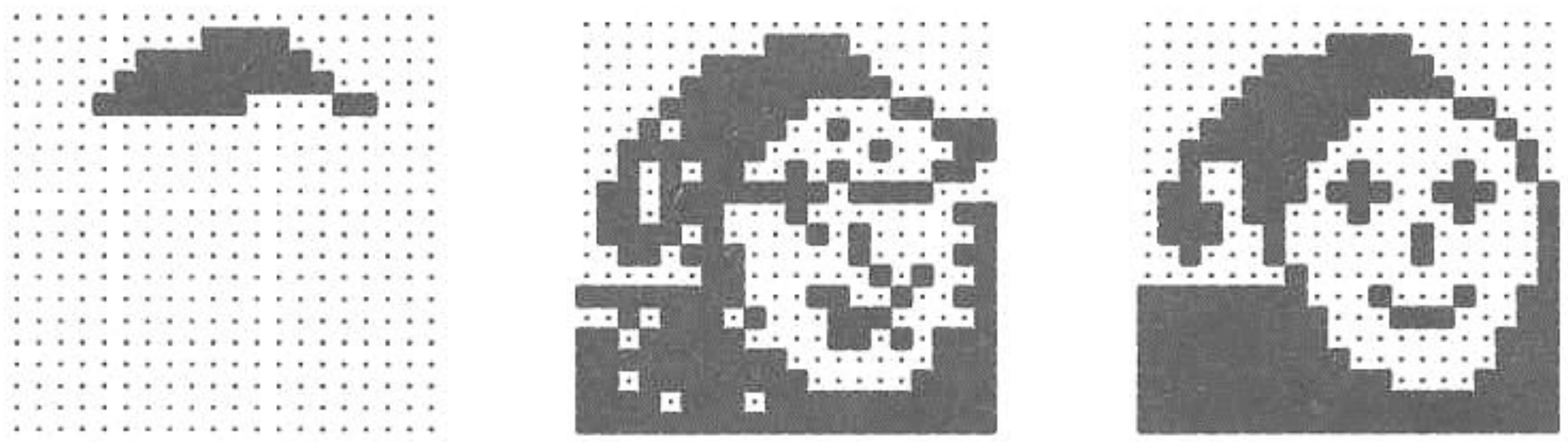
- **Image reconstruction.**
- A 20 X 20 discrete Hopfield network was trained with 20 input patterns, including the one shown in the left figure and 19 random patterns as the one on the right.





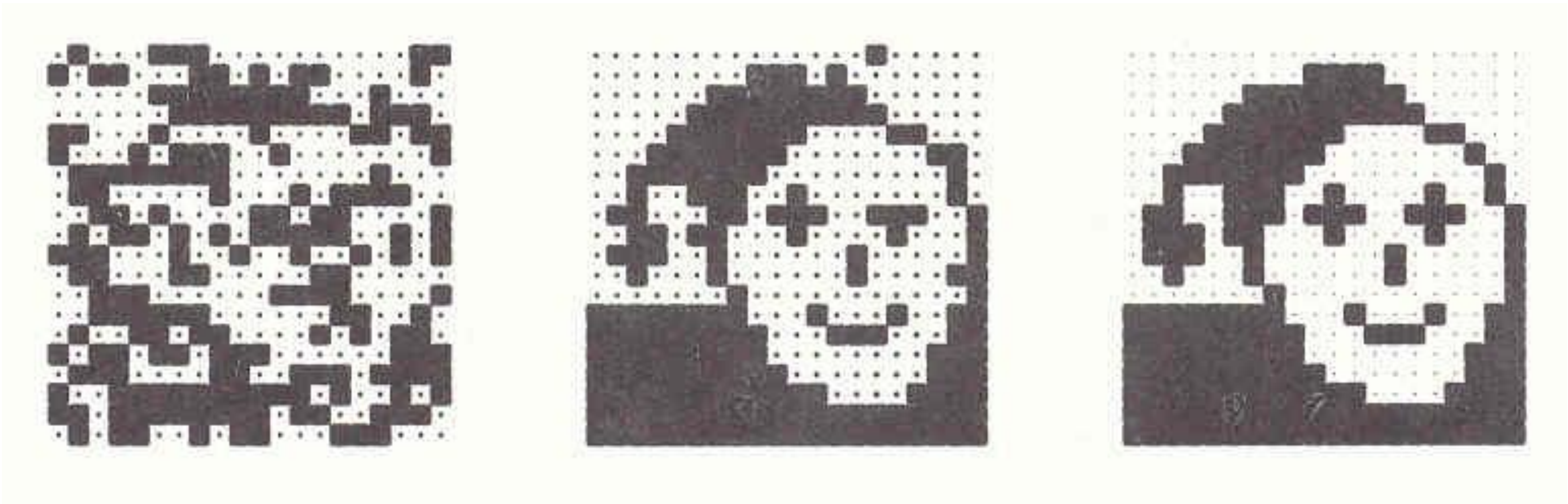
Hopfield Neural network - example

The Hopfield Network 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.



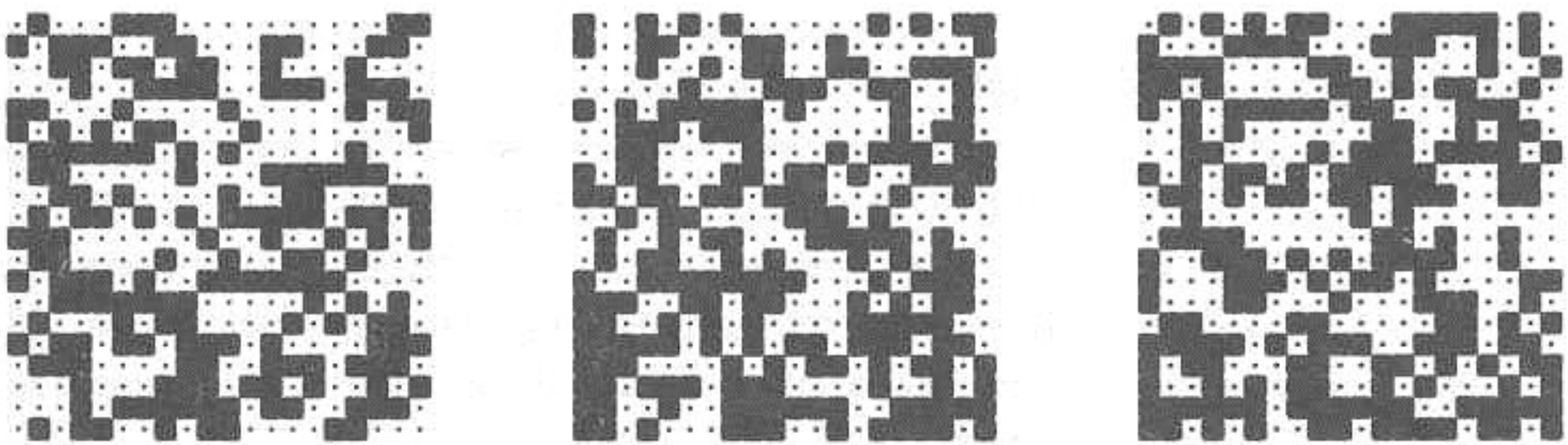
Hopfield Neural network - example

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.



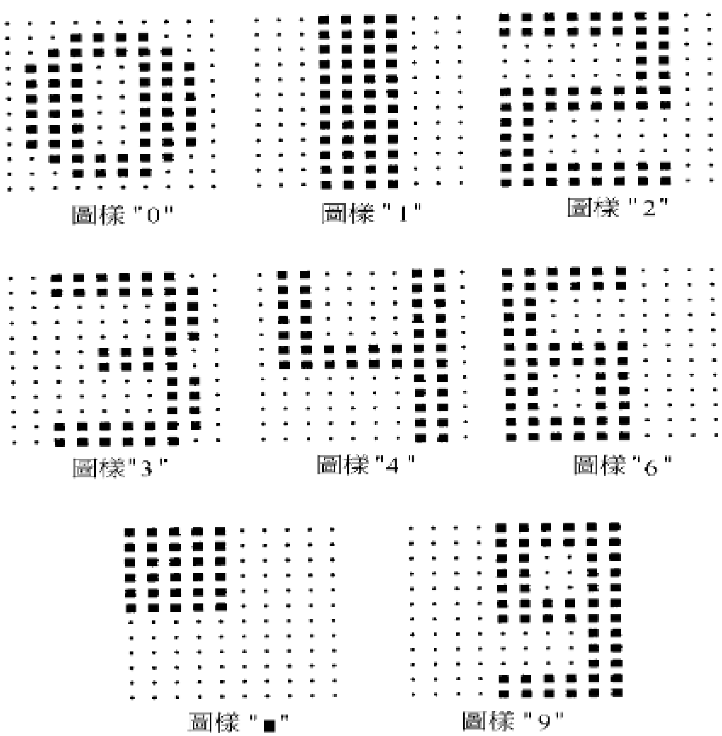
Hopfield Neural network - example

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

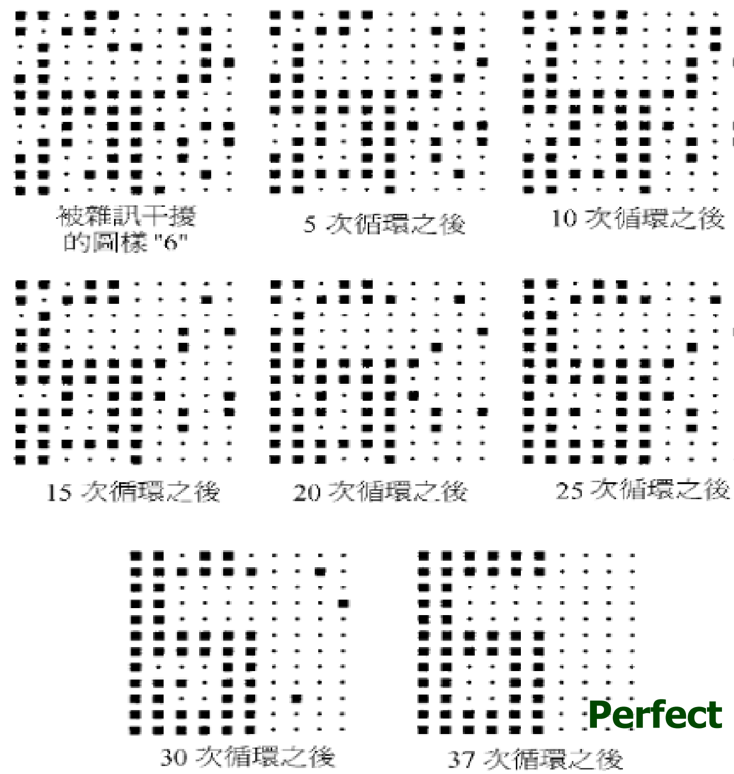


Hopfield Neural network - example

The stored dada in memory



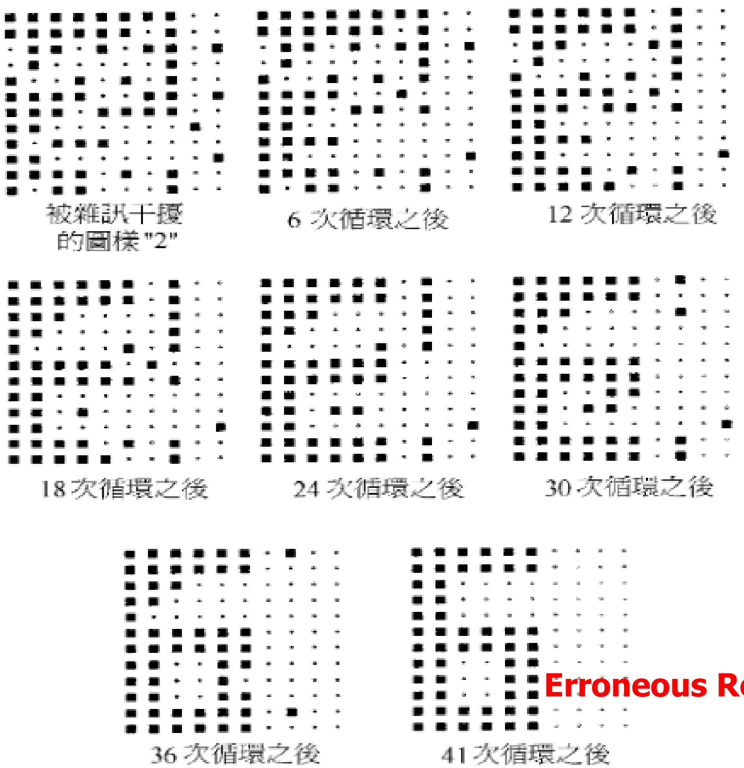
Network input: Distorted 6



Perfect Recall

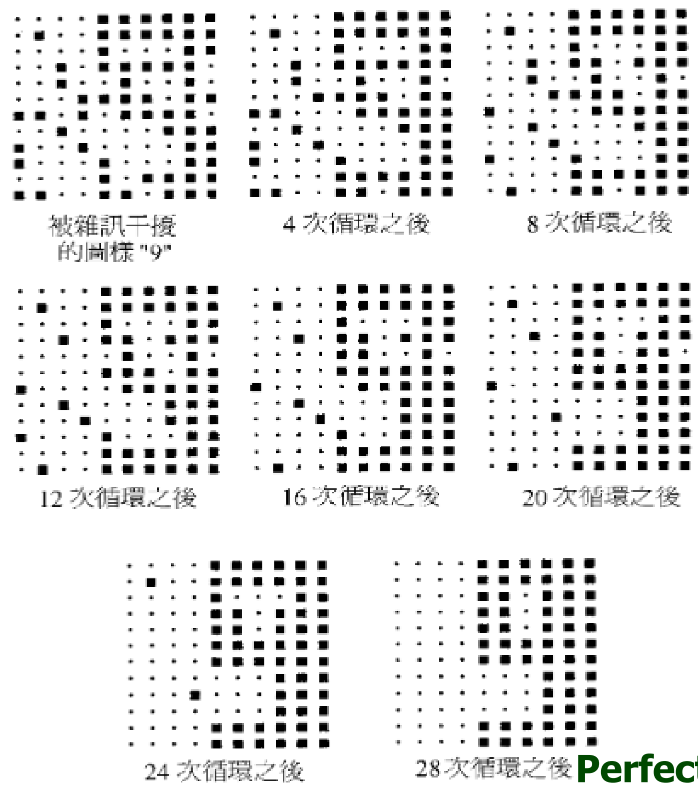
Hopfield Neural network - example

Network input: Distorted 2



Erroneous Recall

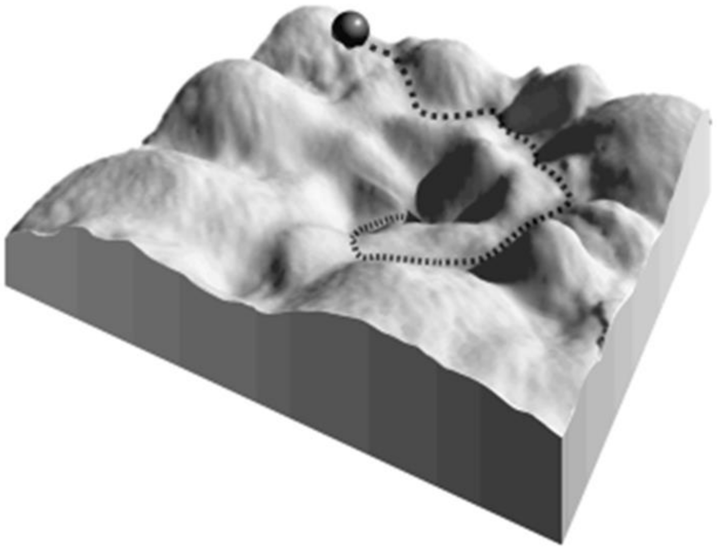
Network input: Distorted 9



Perfect Recall

Hopfield network- Energy function

Hopfield nets have a scalar value associated with each state of the network, referred to as the "energy", E, of the network, where:



$$E = -0.5 \sum_{i \neq j} \sum_{j} y_i y_j w_{ij} + \sum_{i} \theta_i y_i$$

$$\Delta E = -\left[\sum_{j} y_j w_{ij} - \theta_i \right] \Delta y_i$$

$$\Delta E < 0$$

a Hopfield network constantly decreases its energy

Hopfield network- example

Problem Statement

- We need to store a **fundamental pattern (memory)** given by the vector $O = [1, 1, 1, -1]^T$ in a four node binary Hopefield network.
- Presume that the threshold parameters are all equal to zero.

Establishing Connection Weights

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

Hopfield network- example

Network' States and Their Code: Total number of states = 16

State	Code			
A	1	1	1	1
B	1	1	1	-1
C	1	1	-1	-1
D	1	1	-1	1
E	1	-1	-1	1
F	1	-1	-1	-1
G	1	-1	1	-1
H	1	-1	1	1

State	Code			
I	-1	-1	1	1
J	-1	-1	1	-1
K	-1	-1	-1	-1
L	-1	-1	-1	1
M	-1	1	-1	1
N	-1	1	-1	-1
O	-1	1	1	-1
P	-1	1	1	1

Hopfield network- example

Calculating energy function for all states:
θ=0

$$E = -1/2 \sum_{i=1}^{4} \sum_{j=1}^{4} w_{ij} o_i o_j$$

$$\begin{aligned} E = -1/2(& w_{11} o_1 o_1 + w_{12} o_1 o_2 + w_{13} o_1 o_3 + w_{14} o_1 o_4 + \\ & w_{21} o_2 o_1 + w_{22} o_2 o_2 + w_{23} o_2 o_3 + w_{24} o_2 o_4 + \\ & w_{31} o_3 o_1 + w_{32} o_3 o_2 + w_{33} o_3 o_3 + w_{34} o_3 o_4 + \\ & w_{41} o_4 o_1 + w_{42} o_4 o_2 + w_{43} o_4 o_3 + w_{44} o_4 o_4) \end{aligned}$$

Hopfield network- example

For state A = [O_1, O_2, O_3, O_4] [1 , 1, 1, 1]

$$
\begin{aligned}
E = -1/2(0 &+ (1)(1)(1) + (1)(1)(1) + (-1)(1)(1)+ \\
&(1)(1)(1) + 0 + (1)(1)(1) + (-1)(1)(1)+ \\
&(1)(1)(1) + (1)(1)(1) + 0 + (-1)(1)(1)+ \\
&(-1)(1)(1) + (-1)(1)(1) + (-1)(1)(1) + 0)
\end{aligned}
$$

$$
\begin{aligned}
E = -1/2(0 &+ 1 + 1 - 1+ \\
&1 + 0 + 1 - 1+ \\
&1 + 1 + 0 - 1+ \\
&-1 - 1 - 1 + 0)
\end{aligned}
$$

$$E = -1/2(6 - 6) = 0$$

Hopfield network- example

State	Code				Energy
A	1	1	1	1	0
B	1	1	1	-1	-6
C	1	1	-1	-1	0
D	1	1	-1	1	2
E	1	-1	-1	1	0
F	1	-1	-1	-1	2
G	1	-1	1	-1	0
H	1	-1	1	1	2
I	-1	-1	1	1	0
J	-1	-1	1	-1	2
K	-1	-1	-1	-1	0
L	-1	-1	-1	1	-6
M	-1	1	-1	1	0
N	-1	1	-1	-1	2
O	-1	1	1	-1	0
P	-1	1	1	-1	2

Similarly, we can compute the energy level of the other states.

Minimum energy (Stable states)

Hopfield network- example

State Transition for State J = [−1 , −1 , 1 , −1]

Transition 1 (o_1)

$$o_1 = sgn(\sum_{j=1}^{4} w_{ij} o_j - \theta_i) = sgn(w_{12} o_2 + w_{13} o_3 + w_{14} o_4 - 0)$$
$$= sgn((1)(-1) + (1)(1) + (-1)(-1))$$
$$= sgn(+1)$$
$$= +1$$

As a result, the first component of the state J changes from −1 to 1. In other words, the state J transits to the state G

$$J = [-1, -1, 1, -1]^T \ (2) \rightarrow G = [1, -1, 1, -1]^T \ (0)$$

Hopfield network- example

Transition 2 (o_2)

$$o_2 = sgn(\sum_{j=1}^{4} w_{ij} o_j - \theta_i) = sgn(w_{21} o_1 + w_{23} o_3 + w_{24} o_4)$$

$$= sgn((1)(1) + (1)(1) + (-1)(-1))$$

$$= sgn(+3)$$

$$= +1$$

As a result, the second component of the state G changes from -1 to 1. In other words, the state G transits to the state B

B= [1, **1**, 1, -1]

Hopfield network- example

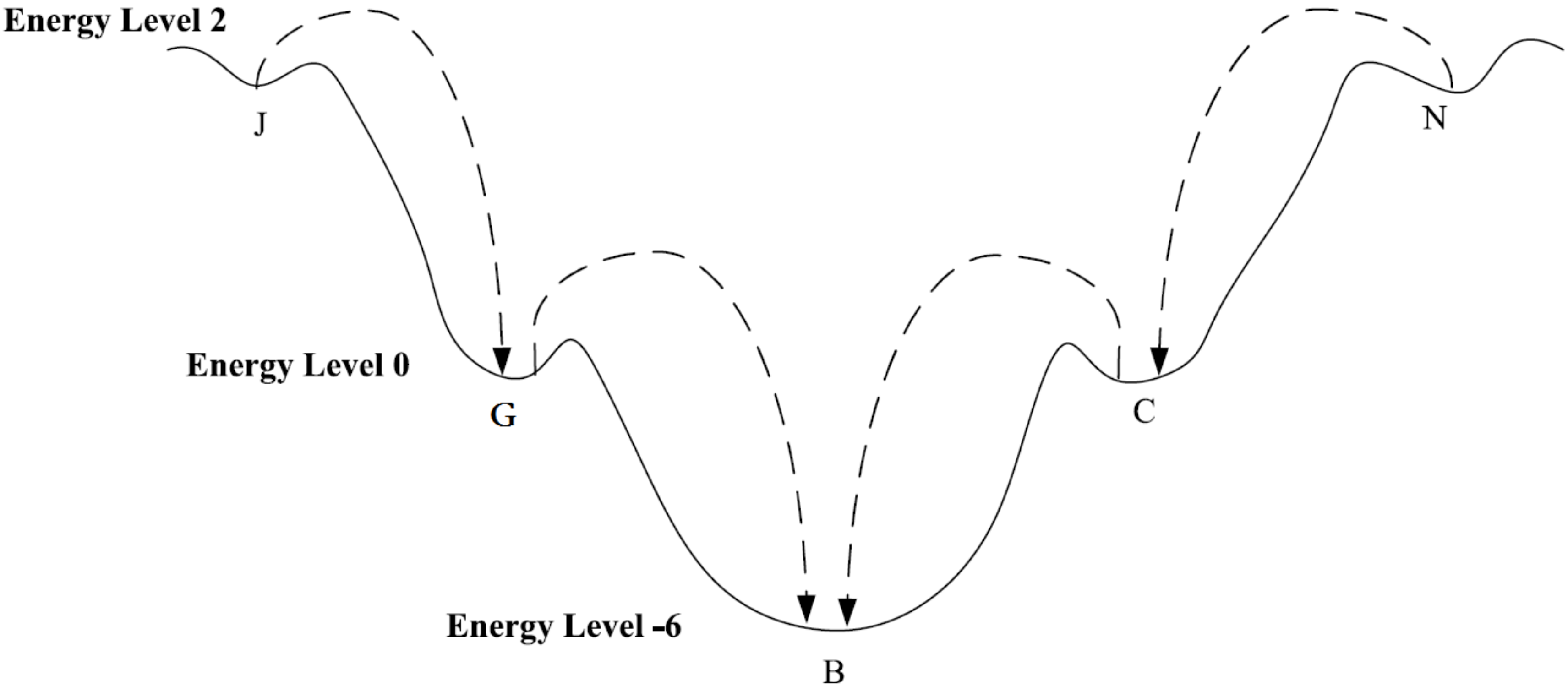
As state B is a fundamental pattern, no more transition will occur

$$o_3 = sgn(\sum_{j=1}^{4} w_{ij} o_j - \theta_i) = sgn(w_{31} o_1 + w_{32} o_2 + w_{34} o_4)$$
$$= sgn((1)(1) + (1)(1) + (-1)(-1))$$
$$= sgn(+3)$$
$$= +1$$

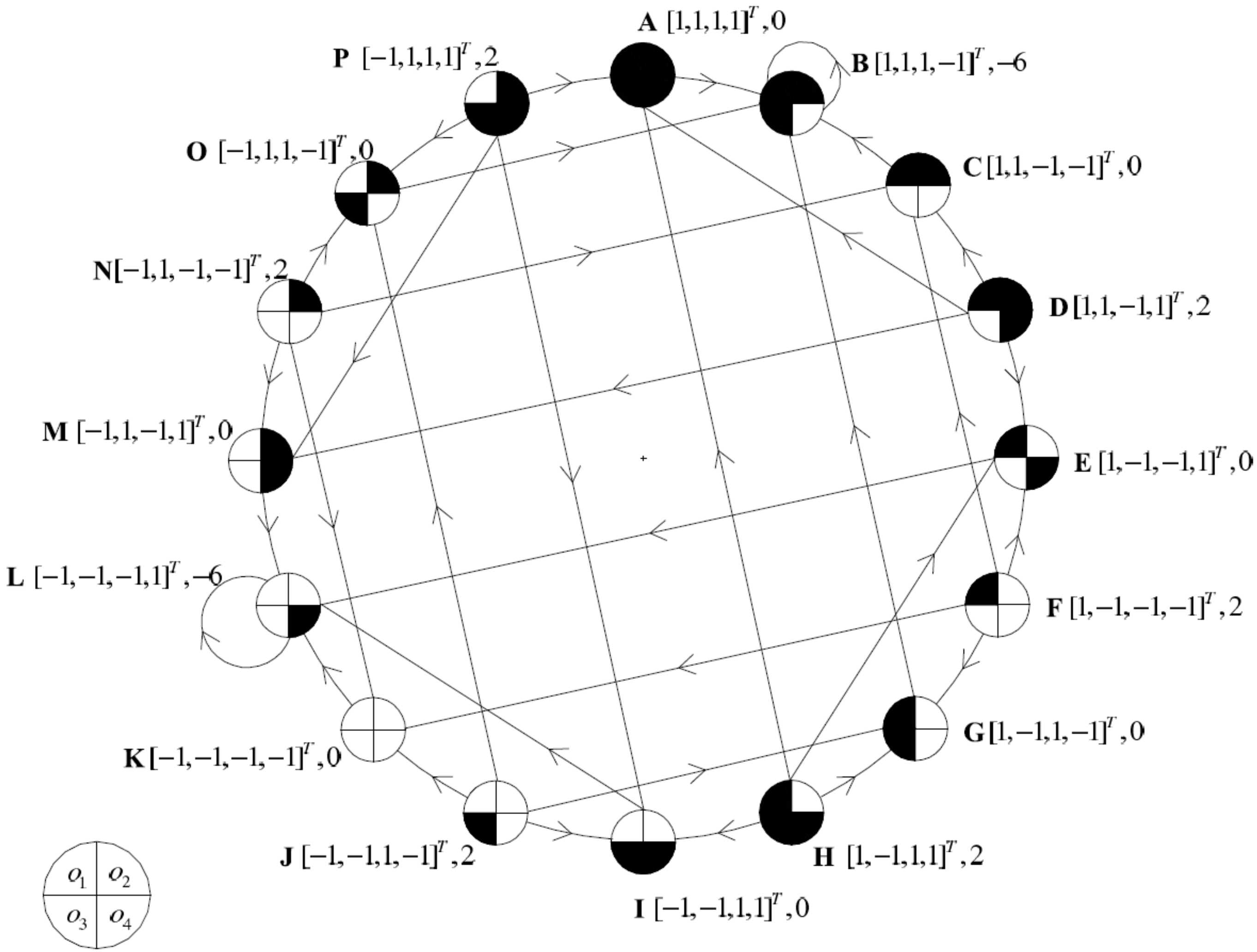
$$o_4 = sgn(\sum_{j=1}^{4} w_{ij} o_j - \theta_i) = sgn(w_{41} o_1 + w_{42} o_2 + w_{43} o_3)$$
$$= sgn((-1)(1) + (-1)(1) + (-1)(1))$$
$$= sgn(-3)$$
$$= -1$$

$$B = [1, 1, 1, -1]^T \ (-6) \rightarrow B = [1, 1, 1, -1]^T \ (-6)$$

Hopfield network- example



Hopfield network- example



Storage Capacity of Hopfield Net

- Binary

$$P \approx 0.15n$$

- Bipolar

$$P \approx \frac{n}{2\log_2 n}$$

P: # of patterns that can be stored an recalled in a net with reasonable accuracy

n: # of neurons in the net


Questions