# AUTOASSOCIATIVE NET

**Used to recall a pattern by a its noisy or incomplete version.(pattern completion/pattern recovery)**

- ***For an auto associative net, the training input and target output are identical i.e.*** $t(p) = s(p)$

- ***Process of training is often called STORING the vectors, may be binary or bipolar [Storage Phase]***

- ***A stored vector can be retrieved from distorted or partial (noisy) input, if input is sufficiently similar to it.[Recall Phase]***

**Associative network can recognize as ”known “ vectors those vectors which are similar to stored vectors, but differ slightly from it.**

***Differences*** **could be two forms:**

1. **MISTAKES**
2. **MISSING data**

**Bipolar (1/-1) representation of inputs and targets**

***Missing Data***: **this is represented by a 0.**
***Mistaken Data***: A mistake in **+1 is represented as -1 and vise versa.**

**Binary(0/1) representation of inputs and targets**
***Missing Data***: **Cannot be represented.**
***Mistaken Data***: A mistake of 1 is represented as 0 and vise versa.

***In general a net can handle more missing components than mistaken/wrong components.***

An Associative net to store **one vector** : recognizing the stored vector

**Vector s = ( 1,1,1,-1) is stored with weight matrix $[s^T s]$**

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

**Diagonal elements not forced to zero**

**Bipolar activation function with threshold as zero**

$$(1,1,1,-1) \cdot W = (4,4,4,-4) \rightarrow (1,1,1,-1)$$

## Testing an autoassociative network with one mistake in (1,1,1,-1)

(1,1,1,-1) ➔ (-1,1,1,-1).W=(2,2,2,-2) ➔ (1,1,1,-1)

(1,1,1,-1) ➔ (1,-1,1,-1).W=(2,2,2,-2) ➔ (1,1,1,-1)

(1,1,1,-1) ➔ (1,1,-1,-1).W=(2,2,2,-2) ➔ (1,1,1,-1)

(1,1,1,-1) ➔ (1,1,1,1).W=(2,2,2,-2) ➔ (1,1,1,-1)

**RECOGNIZES !!**

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

## TESTING AN AUTOASSOCIATIVE NETWORK WITH TWO MISTAKES IN (1,1,1,-1)

(1,1,1,-1) ➔ (-1,-1,1,-1).W=(0,0,0,0) ➔ (1,1,1,1)

**Does not recognize**

# Testing an Autoassociative Network with two "missing" entries in input vector

(1,1,1,-1) → (0,0,1,-1).W=(2,2,2,-2) → (1,1,1,-1)

(1,1,1,-1) → (0,1,0,-1).W=(2,2,2,-2) → (1,1,1,-1)

(1,1,1,-1) → (0,1,1,0).W=(2,2,2,-2) → (1,1,1,-1)

(1,1,1,-1) → (1,0,0,-1).W=(2,2,2,-2) → (1,1,1,-1)

(1,1,1,-1) → (1,0,1,0).W=(2,2,2,-2) → (1,1,1,-1)

(1,1,1,-1) → (1,1,0,0).W=(2,2,2,-2) → (1,1,1,-1)

**Successfully Retrieves**

# (1,1,1,-1)- Stored Vector Two Missing Data

$(1,1,1,-1) \rightarrow (0,0,1,-1).W_0 = (2,2,1,-1) \rightarrow (1,1,1,-1)$

$(1,1,1,-1) \rightarrow (0,1,0,-1).W_0 = (2,1,2,-1) \rightarrow (1,1,1,-1)$

$(1,1,1,-1) \rightarrow (0,1,1,0).W_0 = (2,1,1,-2) \rightarrow (1,1,1,-1)$

$(1,1,1,-1) \rightarrow (1,0,0,-1).W_0 = (1,2,2,-1) \rightarrow (1,1,1,-1)$

$(1,1,1,-1) \rightarrow (1,0,1,0).W_0 = (1,2,1,-2) \rightarrow (1,1,1,-1)$

$(1,1,1,-1) \rightarrow (1,1,0,0).W_0 = (1,1,2,-2) \rightarrow (1,1,1,-1)$

RECOGNIZES IN ALL CASES

# AN AUTOASSOCIATIVE NETWORK WITH NO SELF-CONNECTIONS :ZEROING OUT DIAGONAL OF ORIGINAL W

$$W = \begin{bmatrix} 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & -1 \\ -1 & -1 & -1 & 1 \end{bmatrix} \qquad W_0 = \begin{bmatrix} 0 & 1 & 1 & -1 \\ 1 & 0 & 1 & -1 \\ 1 & 1 & 0 & -1 \\ -1 & -1 & -1 & 0 \end{bmatrix}$$

**Testing an Autoassociative network with two mistakes in (1,1,1,-1)**

(1,1,1,-1) ➔ (-1,-1,1,-1).W=(0,0,0,0) ➔ (0,0,0,0)

(1,1,1,-1) ➔ (-1,-1,1,-1).$W_0$=(1,1,1,1) ➔ (1,1,1,1)

Still does not recognize

# STORING SEVERAL VECTORS IN AN AUTO-ASSOCIATIVE NETWORK

***Non-orthogonal vectors cannot be stored in an auto-associative network in reliable manner.*** **They may not always be identified.**

**More patterns can be stored if they are not similar to each other (e.g., orthogonal)**

Two vectors x and y are orthogonal if

$$x\,y^T = 0 \text{ i.e., } \Sigma\, x_i\, y_i = 0$$

## Example: Storing more than one vector in an auto-associative net

- **More than one vector can be stored in an auto-associative net by simply adding the weights needed for each vector.**

***Theorem: an n x n network is able to store up to n-1 mutually orthogonal (M.O.) bipolar vectors , but not n such vectors.***

**The requirement of orthogonality places serious limitations on the Hebbian Learning Rule**

# Example: Storing 2 non-orthogonal vectors in an auto-associative net

Trying to store two non-orthoganal vectors (1,-1,-1,1) and (1,1,-1,1) gives a weight matrix which cannot distinguish between two vectors

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

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

# TESTING

$$[\,1, 1, -1, 1\,] \begin{bmatrix} 0 & 0 & -2 & 2 \\ 0 & 0 & 0 & 0 \\ -2 & 0 & 0 & -2 \\ 2 & 0 & -2 & 0 \end{bmatrix} = [4,0,-4,4] = [\,1,-1,-1,1\,]$$

$$[\,1, -1, -1, 1\,] \begin{bmatrix} 0 & 0 & -2 & 2 \\ 0 & 0 & 0 & 0 \\ -2 & 0 & 0 & -2 \\ 2 & 0 & -2 & 0 \end{bmatrix} = [4,0,-4,4] = [1,-1,-1,1\,]$$

**Same result for both, cannot distinguish between the two vectors**

# Example : store (1 1 -1 -1) and (-1 1 1 -1) [orthogonal] in an auto-associative net.

- to store (1 1 -1 -1), we need a weight matrix $\mathbf{W}_1$ and to store (-1 1 1 -1), we need a weight matrix $\mathbf{W}_2$.

$$\mathbf{W}_1 = \begin{matrix} 0 & 1 & -1 & -1 \\ 1 & 0 & -1 & -1 \\ -1 & -1 & 0 & 1 \\ -1 & -1 & 1 & 0 \end{matrix} \qquad \mathbf{W}_2 = \begin{matrix} 0 & -1 & -1 & 1 \\ -1 & 0 & 1 & -1 \\ -1 & 1 & 0 & -1 \\ 1 & -1 & -1 & 0 \end{matrix}$$

To store (1 1 -1 -1) and (-1 1 1 -1), we simply add $\mathbf{W}_1$ and $\mathbf{W}_2$ to obtain the new weight matrix W. This is because the two vectors are orthogonal.

# Storing 3 mutually orthogonal vectors in an auto-associative net

Let **W1** + **W2** be the weight matrix to store the orthogonal vectors

(1, 1, − 1, − 1) and (− 1, 1, 1, − 1)

**W3** be the weight matrix that stores (− 1, 1, − 1, 1)

## $X_3$ is orthogonal to both $X_1$ and $X_2$

$$\underset{\mathbf{W}_1 + \mathbf{W}_2}{\begin{bmatrix} 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & -2 \\ -2 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 \end{bmatrix}} + \underset{\mathbf{W}_3}{\begin{bmatrix} 0 & -1 & 1 & -1 \\ -1 & 0 & -1 & 1 \\ 1 & -1 & 0 & -1 \\ -1 & 1 & -1 & 0 \end{bmatrix}} = \underset{\mathbf{W}_1 + \mathbf{W}_2 + \mathbf{W}_3}{\begin{bmatrix} 0 & -1 & -1 & -1 \\ -1 & 0 & -1 & -1 \\ -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 \end{bmatrix}}$$

which correctly classifies each of the three vectors on which it was trained.

# Storing 4 mutually orthogonal vectors in an auto-associative net

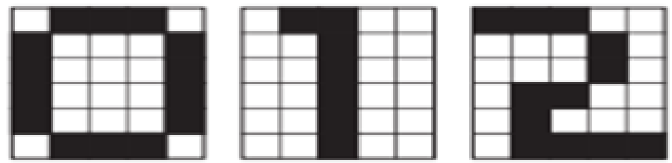
Attempting to store a fourth vector, (1, 1, 1, 1), with weight matrix W4 orthogonal to each of the foregoing three,

$$
\begin{array}{ccccc}
\mathbf{W}_1 + \mathbf{W}_2 + \mathbf{W}_3 & & \mathbf{W}_4 & & \mathbf{W}^* \\
\begin{bmatrix} 0 & -1 & -1 & -1 \\ -1 & 0 & -1 & -1 \\ -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 \end{bmatrix} & + & \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} & = & \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}
\end{array}
$$

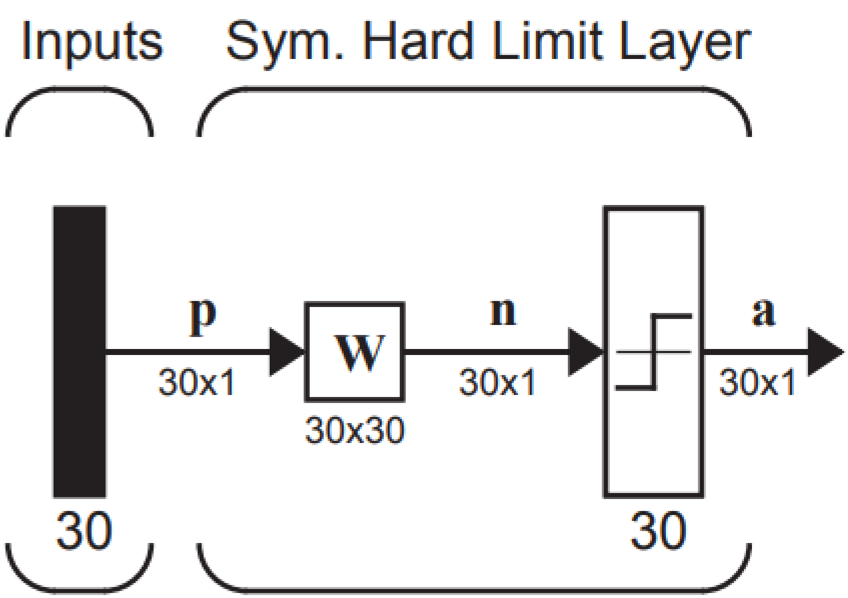
which cannot recognize any vector

# Autoassociative Memory

**pattern and target same 6x5 matrix**

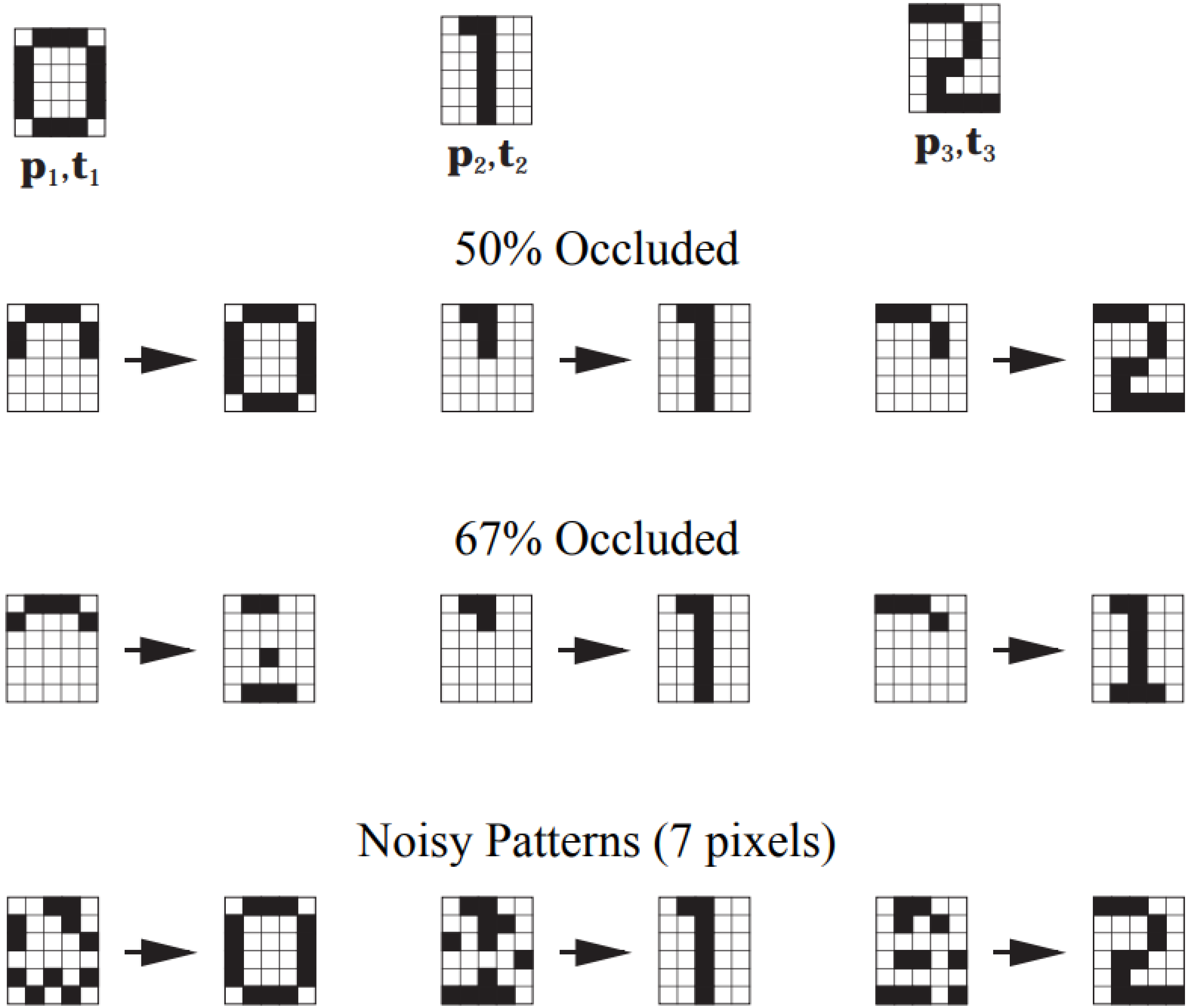


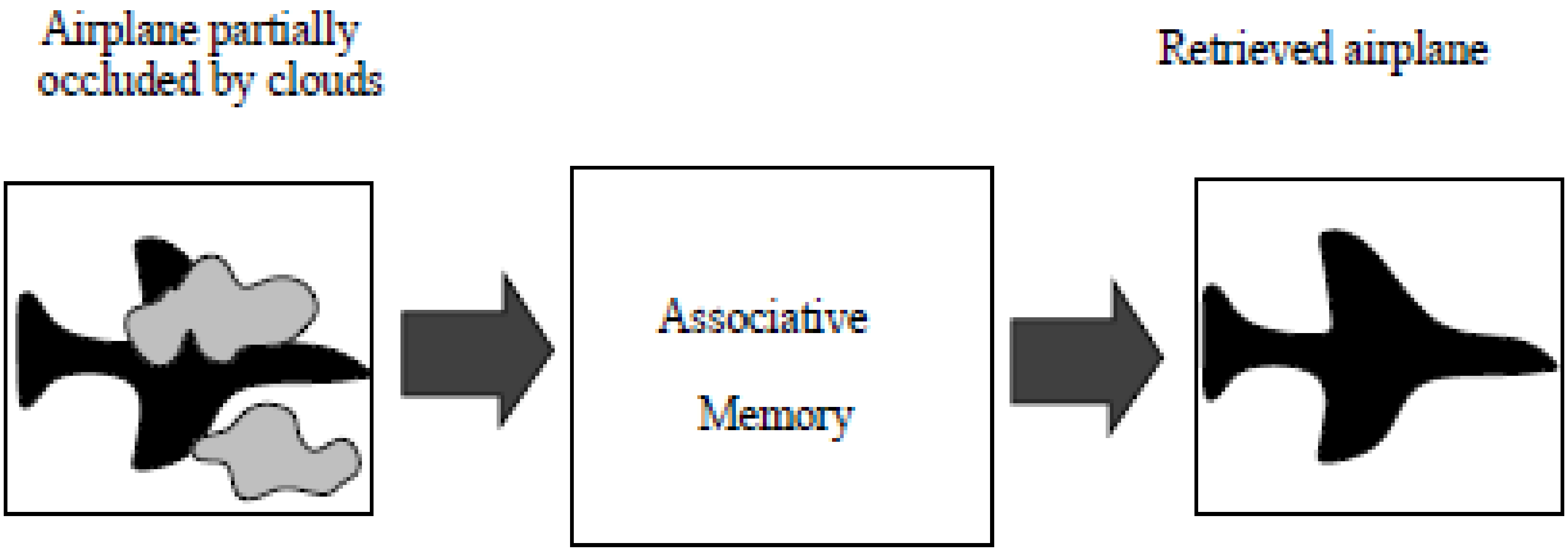
$\mathbf{p}_1,\mathbf{t}_1$ $\mathbf{p}_2,\mathbf{t}_2$ $\mathbf{p}_3,\mathbf{t}_3$




$\mathbf{a} = \mathbf{hardlims}(\mathbf{Wp})$

$$\mathbf{W} = \mathbf{p}_1\mathbf{p}_1^T + \mathbf{p}_2\mathbf{p}_2^T + \mathbf{p}_3\mathbf{p}_3^T$$


$\mathbf{p}_1, \mathbf{t}_1$
$\mathbf{p}_2, \mathbf{t}_2$
$\mathbf{p}_3, \mathbf{t}_3$
50% Occluded
67% Occluded
Noisy Patterns (7 pixels)


Airplane partially occluded by clouds
Associative
Memory
Retrieved airplane

END