Associative Memories

Berlin Chen, 2002

Introduction

- **Associative memories**: Systems for associating the input patterns with the stored patterns (prototypes)
  - **Dynamic systems** with feedback networks
  - **Static/Feedforward systems** without feedback networks

- **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

Introduction

- No usable addressing scheme exists
  - Memory information spatially distributed and superimposed through the network
  - No memory locations have addresses
- Expectations regarding associative memories
  - As large of a capacity of *P* stored patterns as possible
  - Data to be stored in a robust manner
  - Adaptability: Addition or elimination of associations

Basic Concepts

- Associative Mapping of Inputs to Outputs

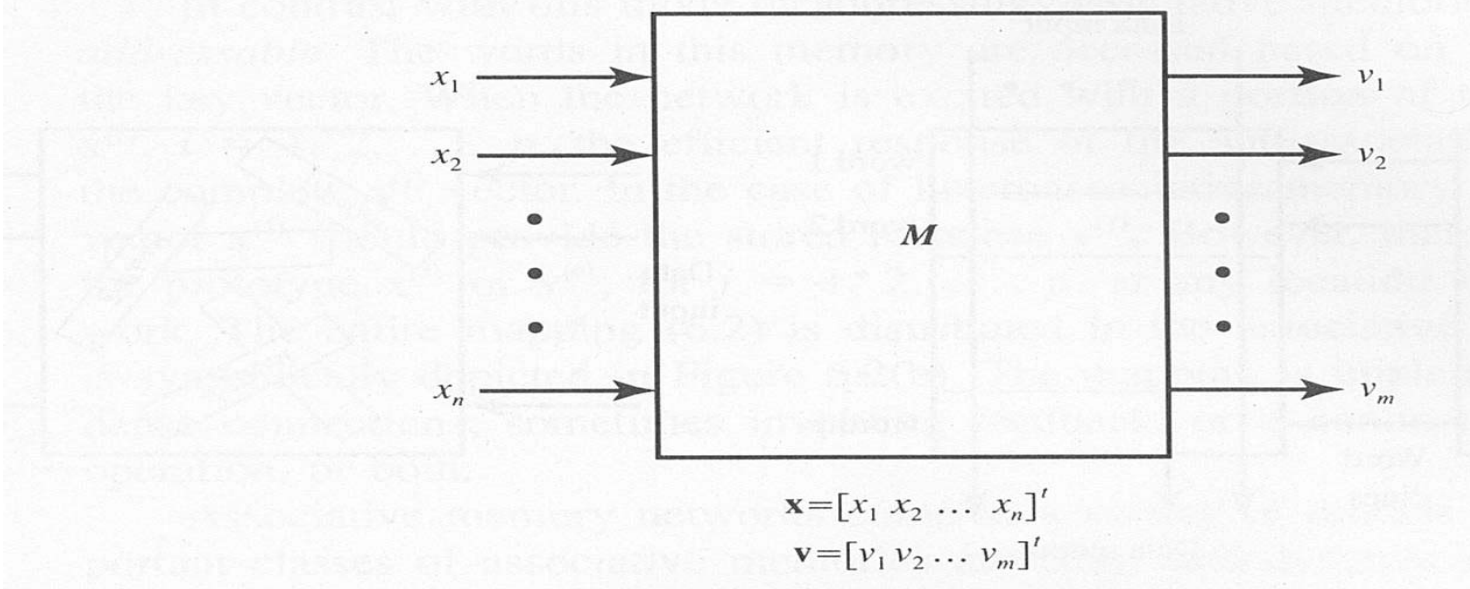


Figure 6.1 Block diagram of an associative memory.

$$\text{Retreival}: \quad \boldsymbol{v} = M\left[\boldsymbol{x}\right]$$

Basic Concepts

- *M* is a general matrix-type operator
  - Memory Paradigms
    - Dynamic systems
    - Static/Feedforward systems

  - Recording and Retrieval
    - **Recording**
      - *M* is expressed as the prototype vectors stored
    - **Retrieval**
      - Mapping: $\boldsymbol{x} \rightarrow \boldsymbol{v}$
      - Linear or nonlinear
      - Input a key vector ***x*** and find a desired vector ***v*** previously stored in the network

Basic Concepts

- Input/Output Relation
  - Heteroassociative memory

Two sets of prototype vectors

$$\boldsymbol{x}^{(i)} \rightarrow \boldsymbol{v}^{(i)}\Big|_{\boldsymbol{v}^{(i)} \neq \boldsymbol{x}^{(i)}} \quad \text{for} \quad i = 1,\ldots,p$$

    - With same dimensionality or not

  - Autoassociative Memory

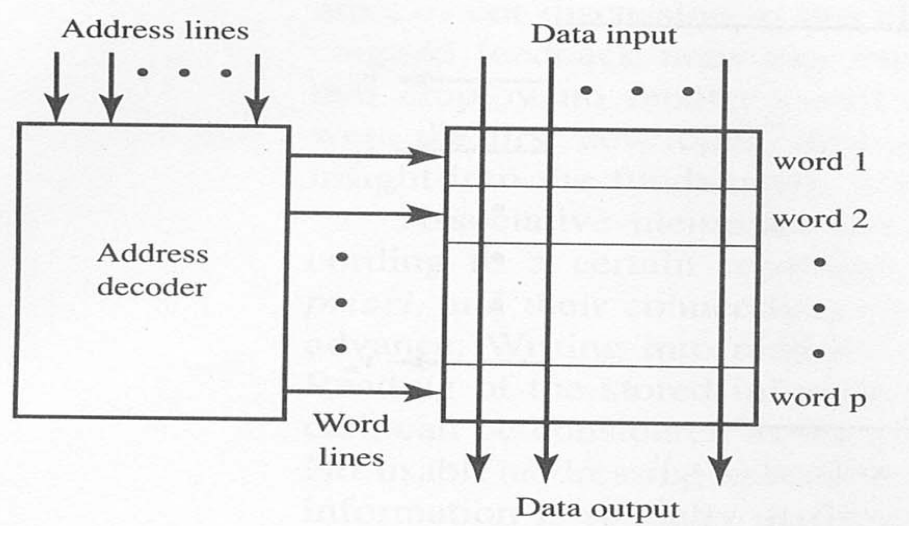
One set of prototype vectors

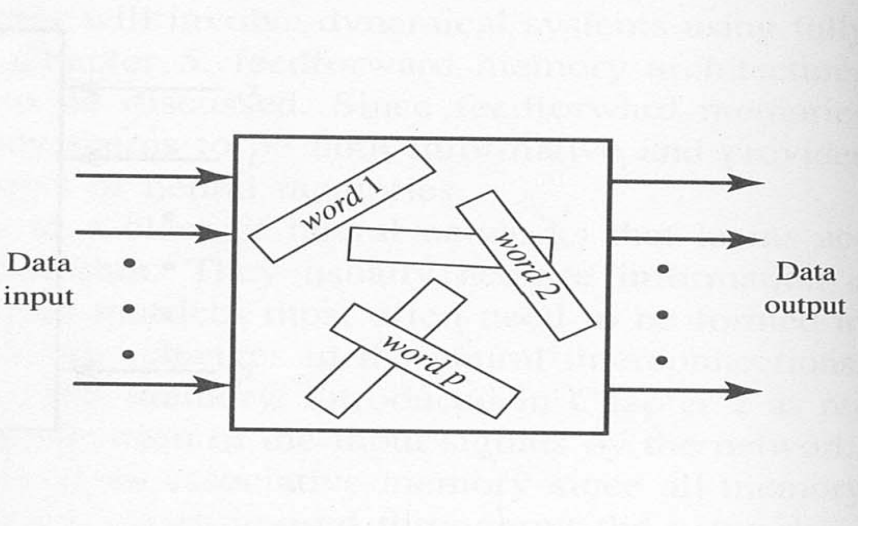
$$\boldsymbol{x}^{(i)} \rightarrow \boldsymbol{v}^{(i)}\Big|_{\boldsymbol{v}^{(i)} = \boldsymbol{x}^{(i)}} \quad \text{for} \quad i = 1,\ldots,p$$

Most for recovery of undistorted prototypes

Basic Concepts

- Comparison of two common addressing modes for memory data retrieval



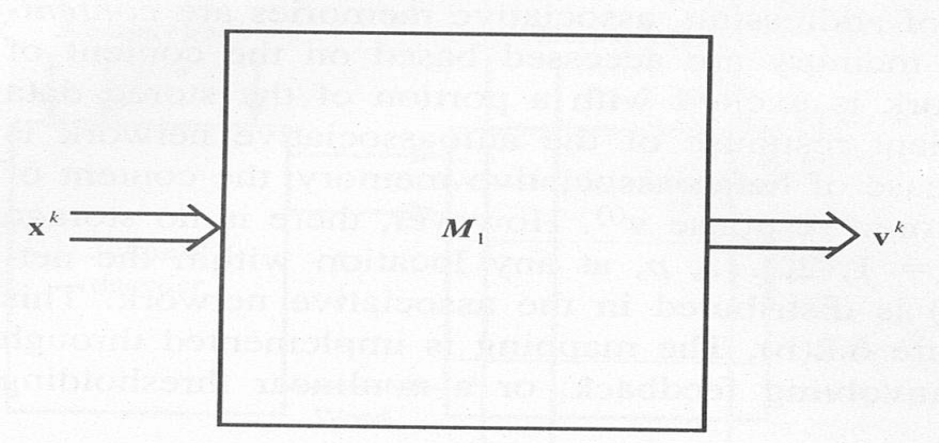


Address-addressable memory

Content-addressable memory

Basic Concepts

- Static Memory
  - Feedforward
  - Non-recurrent (no feedback, no delay), instantaneous

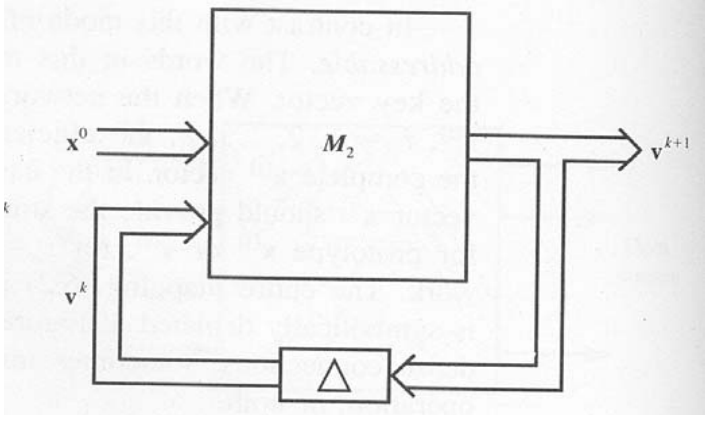


$$\boldsymbol{v}^k = M_1\left[\boldsymbol{x}^k\right]$$

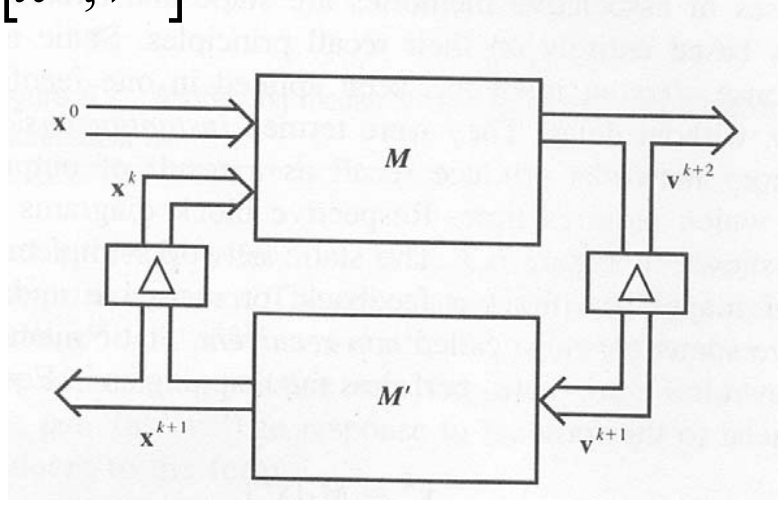
Basic Concepts

- Dynamic Memory
  - Recall as a result of output/input feedback interactions
  - Recurrent, time-delayed
  - Dynamically evolved and finally converged to an equilibrium state

$$\boldsymbol{v}^{k+1} = M_2\left[\boldsymbol{x}^k, \boldsymbol{v}^k\right]$$



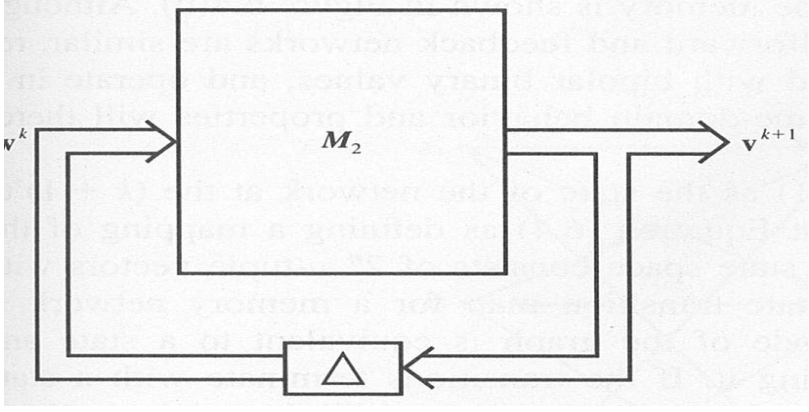
Recurrent Autoassociative Memory



Recurrent Heteroassociative Memory

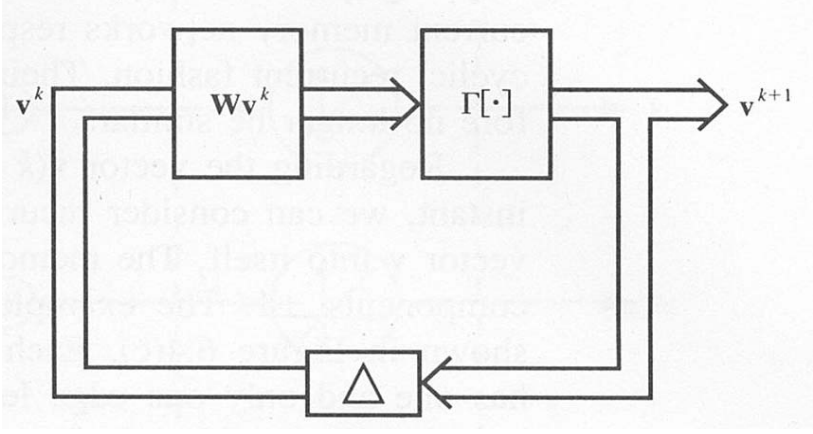
Basic Concepts

- Dynamic Memory
  - **Hopfield model**: an recurrent network for which the input $\boldsymbol{x}^0$ is used to initialize $\boldsymbol{v}^0$, i.e. $\boldsymbol{x}^0 = \boldsymbol{v}^0$, and the input is then removed for the following evolution



$$\boldsymbol{v}^{k+1} = M_2\left[\boldsymbol{v}^k\right]$$

nonlinear mapping



$$\boldsymbol{v}^{k+1} = \Gamma\left[\boldsymbol{W}\boldsymbol{v}^k\right]$$

Basic Concepts

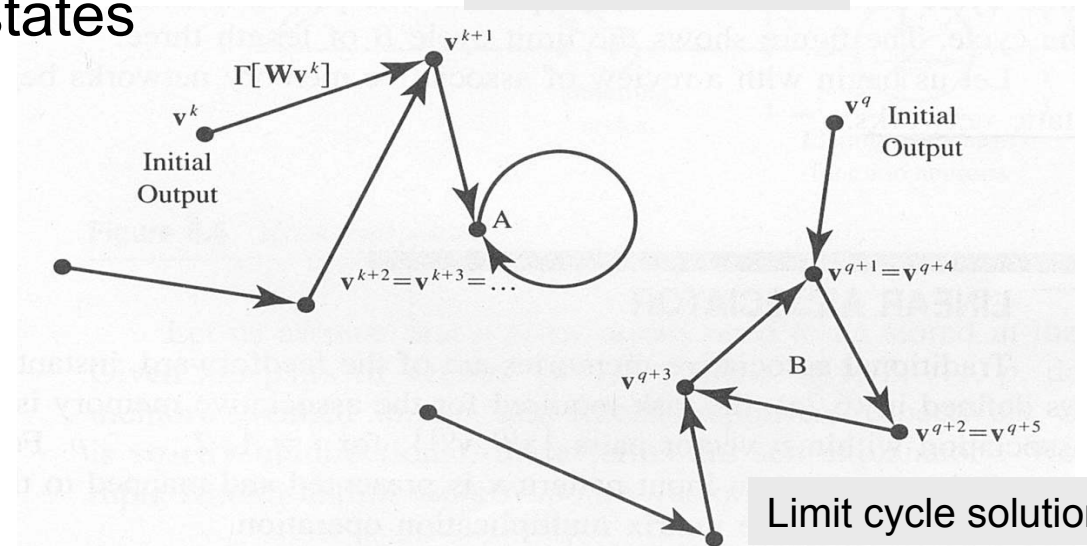
- Dynamic Memory
  - **Hopfield model**:
    - Operated on bipolar binary values (±1)
    - Hard-limiting (binary) activation function

$$\Gamma = \begin{bmatrix} \text{sgn}(\cdot) & 0 & \cdots\cdots & 0 \\ 0 & \text{sgn}(\cdot) & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots\cdots & \cdots & \text{sgn}(\cdot) \end{bmatrix}$$

$$v_i = f(net_i) = \text{sgn}(net_i)$$

  - Equilibrium states

Fixed point solution



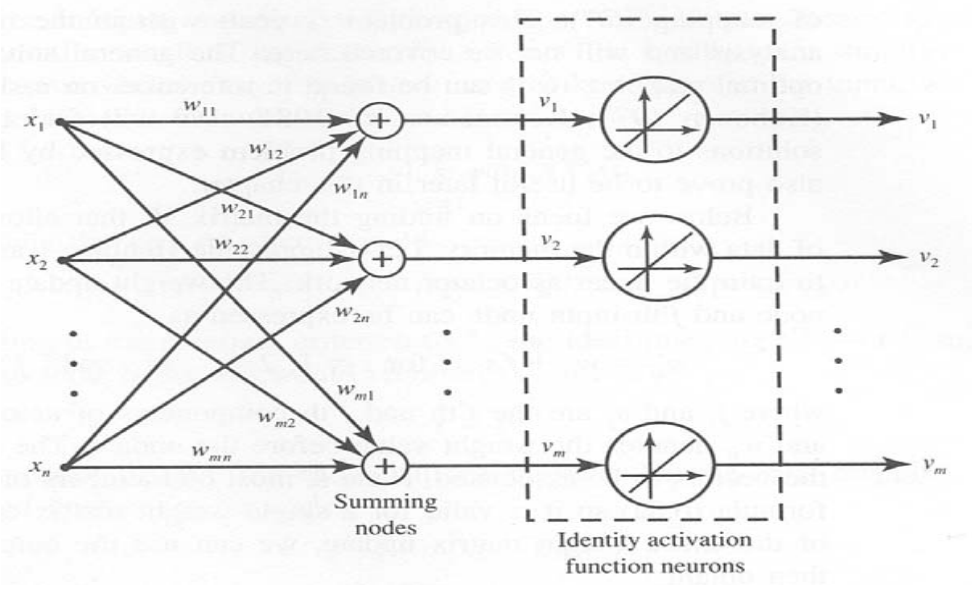
Limit cycle solution

Linear Associative Memory

- Belong to Static Networks
- No nonlinear or delay operations
- Dummy neurons with identity activation functions

$$v_i = f(net_i) = net_i$$

$$\boldsymbol{v} = \boldsymbol{W}\boldsymbol{x}$$



Linear Associative Memory

- Given *p* associations $\{\boldsymbol{s}^{(i)}, \boldsymbol{f}^{(i)}\}$

$$\boldsymbol{s}^{(i)} = \left[s_1^{(i)}, s_2^{(i)}, \ldots, s_n^{(i)}\right]^t$$ **Inputs** **Stimuli**

$$\boldsymbol{f}^{(i)} = \left[f_1^{(i)}, f_2^{(i)}, \ldots, f_m^{(i)}\right]^t$$ **Outputs** **Forced response**

- Apply Hebbian Learning Rule

$$w'_{ij} = w_{ij} + f_i s_j$$

$$\Rightarrow \boldsymbol{W}' = \boldsymbol{W} + \boldsymbol{f}\boldsymbol{s}^t$$ outer product

$$\boldsymbol{W}_0 = \boldsymbol{0}$$

$$\Rightarrow \boldsymbol{W}' = \boldsymbol{f}^{(i)}\boldsymbol{s}^{(i)t}$$

$$\boldsymbol{F} \overset{\Delta}{=} \left[\boldsymbol{f}^{(1)} \; \boldsymbol{f}^{(2)} \; \ldots \; \boldsymbol{f}^{(p)}\right]$$

$$\boldsymbol{S} \overset{\Delta}{=} \left[\boldsymbol{s}^{(1)} \; \boldsymbol{s}^{(2)} \; \ldots \; \boldsymbol{s}^{(p)}\right]$$

p pairs learned

$$\Rightarrow \boldsymbol{W}' = \sum_{i=1}^{p} \boldsymbol{f}^{(i)}\boldsymbol{s}^{(i)t} \Rightarrow \boldsymbol{W}' = \boldsymbol{F}\boldsymbol{S}^t$$

Cross-correlation matrix: m x n

Hebbian Learning

Hebb, 1949

- A purely feedforward, unsupervised learning
- The learning signal is equal to the neuron's output

$$r = o_i = f\left(\mathbf{w}_i^t \mathbf{x}\right)$$

$$\Delta \mathbf{w}_i = c \cdot o_i \cdot \mathbf{x} = cf\left(\mathbf{w}_i^t \mathbf{x}\right) \cdot \mathbf{x} \quad \textbf{or } \Delta w_{ij} = c \cdot f\left(\mathbf{w}_i^t \mathbf{x}\right) \cdot x_j$$

- The weight initialization at small random values around $\mathbf{w}_i$=0 prior to learning
- If the crossproduct of output and input (or correlation) is positive, it results in an increase of the weight, otherwise the weight decreases

Hebbian Learning (cont.)

- "When an axon of cell A is near enough to excite a cell B and repeatedly or persistently take places in firing it, some growth process or metabolic changes take place in one or both cells such that A's efficiency, as one of the cell firing B, is increased" (Hebb, 1949)

當神經元 A 的軸突與神經元 B 之距離，近到足以激發它的地步時，
若神經元 A 重複地或持續地扮演激發神經元 B 的角色，則某種
增長現象或新陳代謝的改變，會發生在其中之一或兩個神經元的細胞上，
以至於神經元 A 能否激發神經元 B 的有效性會被提高。

The weight connecting to neuron B

參考自蘇木村教授著作

Linear Associative Memory

- Perform the associative recall using $\boldsymbol{s}^{(j)}$

$$\boldsymbol{v} = \boldsymbol{W}\,'\boldsymbol{s}^{(j)} = \left(\sum_{i=1}^{p} \boldsymbol{f}^{(i)} \boldsymbol{s}^{(i)t} \right) \boldsymbol{s}^{(j)}$$

$$= \boldsymbol{f}^{(1)} \boldsymbol{s}^{(1)t} \boldsymbol{s}^{(j)} + \ldots + \boldsymbol{f}^{(j)} \boldsymbol{s}^{(j)t} \boldsymbol{s}^{(j)} + \ldots \boldsymbol{f}^{(p)} \boldsymbol{s}^{(p)t} \boldsymbol{s}^{(j)}$$

  - If the desired response is $\boldsymbol{v} = \boldsymbol{f}^{(j)}$
    - Necessary conditions: **orthonormal**

$$\boldsymbol{s}^{(i)t} \boldsymbol{s}^{(j)} = 0, \quad \text{for} \quad i \neq j$$

$$\boldsymbol{s}^{(j)t} \boldsymbol{s}^{(j)} = 1$$

    - Rather strict and may not always hold

Linear Associative Memory

- If a distorted input presented

$$\boldsymbol{s}^{(j)'} = \boldsymbol{s}^{(j)} + \boldsymbol{\Delta}^{(j)}$$

$$\boldsymbol{v} = \boldsymbol{f}^{(j)} + \boldsymbol{f}^{(j)}\boldsymbol{s}^{(j)t}\boldsymbol{\Delta}^{(j)} + \sum_{i \neq j}^{p} \boldsymbol{f}^{(i)}\boldsymbol{s}^{(i)t}\boldsymbol{\Delta}^{(j)}$$

If $\boldsymbol{s}^{(j)}$ and $\boldsymbol{\Delta}^{(j)}$ statically indenpendent thus orthognal

$$\boldsymbol{v} = \boldsymbol{f}^{(j)} + \sum_{i \neq j}^{p} \boldsymbol{f}^{(i)}\boldsymbol{s}^{(i)t}\boldsymbol{\Delta}^{(j)}$$

Cross-talk noise

  - Cross-talk noise remains additive at the memory output to the originally stored association
  - Linear associative memory perform rather poorly when with distorted stimuli vectors
    - Limited usage

Linear Associative Memory

- Example: three associations $(\boldsymbol{s}^{(1)}, \boldsymbol{f}^{(1)}), (\boldsymbol{s}^{(2)}, \boldsymbol{f}^{(2)}), (\boldsymbol{s}^{(3)}, \boldsymbol{f}^{(3)})$

$$\boldsymbol{s}^{(1)} = (1,0,0)^T, \boldsymbol{s}^{(2)} = (0,1,0)^T, \boldsymbol{s}^{(3)} = (0,0,1)^T$$

$$\boldsymbol{f}^{(1)} = (2,1,2)^T, \boldsymbol{f}^{(2)} = (1,2,3)^T, \boldsymbol{f}^{(3)} = (3,1,2)^T$$

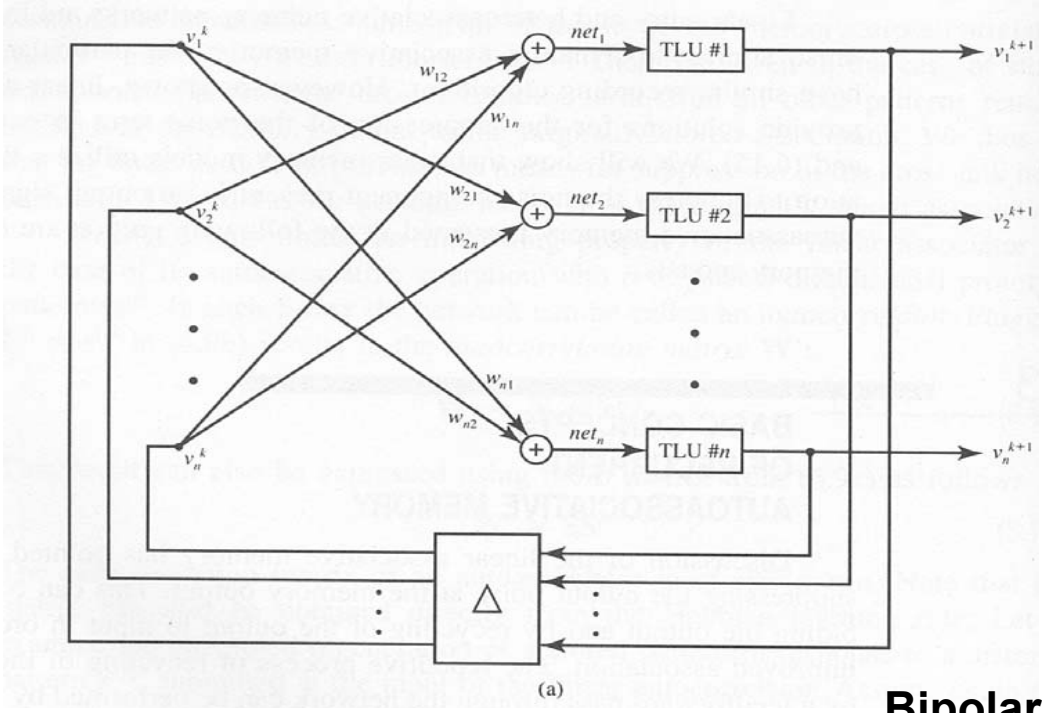
$$M' = \boldsymbol{f}^{(1)}\boldsymbol{s}^{(1)t} + \boldsymbol{f}^{(2)}\boldsymbol{s}^{(2)t} + \boldsymbol{f}^{(3)}\boldsymbol{s}^{(3)t}$$

$$= \begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} + \begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 3 \\ 1 & 2 & 1 \\ 2 & 3 & 2 \end{bmatrix}$$

$$\boldsymbol{v} = M'\boldsymbol{s}^{(2)} = \begin{bmatrix} 2 & 1 & 3 \\ 1 & 2 & 1 \\ 2 & 3 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

Hopfield's Autoassociative Memory (1982,1984)

- Recurrent/Dynamic Associative Memory
- Discrete-time and Asynchronous Update Mode



Bipolar Binary Responses

Hopfield's Autoassociative Memory

- Storage Algorithm

$$\boldsymbol{W} = \left(\sum_{m=1}^{p} \boldsymbol{s}^{(m)} \boldsymbol{s}^{(m)t} \right) - p\boldsymbol{I}$$

$$w_{ij} = \left(1 - \delta_{ij}\right) \sum_{m=1}^{p} s_i^{(m)} s_j^{(m)}, \text{ where } \begin{cases} \delta_{ij} = 1 & \text{if } i = j \\ \delta_{ij} = 0 & \text{if } i \neq j \end{cases}$$

Only for bipolar binary response

  - The diagonal elements set to be zero to avoid self-feedbacks
  - The matrix only represents the correlation terms among the vector entries
  - Invariant with respect to the sequence of storing patterns
  - Additional autoassociations can be added at any time by superimposing, autoassociations also can be removed

Hopfield's Autoassociative Memory

- Retrieval Algorithm
  - The output update rule can be expressed as:

$$v_i^{k+1} = \text{sgn}\left(\sum_{j=1}^{n} w_{ij} v_j^k \right)$$

$$T_i : \text{threshold value of neuron } i$$

  - Asynchronous and random update of neurons *m, p, q*

First Update $\quad \boldsymbol{v}^1 = \left[v_1^0 \; v_2^0 \ldots v_m^1 \ldots v_p^0 \ldots v_q^0 \ldots . v_m^0 \right]$

Second Update $\quad \boldsymbol{v}^2 = \left[v_1^0 \; v_2^0 \ldots v_m^1 \ldots v_p^2 \ldots v_q^0 \ldots . v_m^0 \right]$

Third Update $\quad \boldsymbol{v}^3 = \left[v_1^0 \; v_2^0 \ldots v_m^1 \ldots v_p^2 \ldots v_q^3 \ldots . v_m^0 \right]$

Hopfield's Autoassociative Memory

- **Example**: two autoassociations

$$\boldsymbol{s}^{(1)} = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}, \boldsymbol{s}^{(2)} = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}$$

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

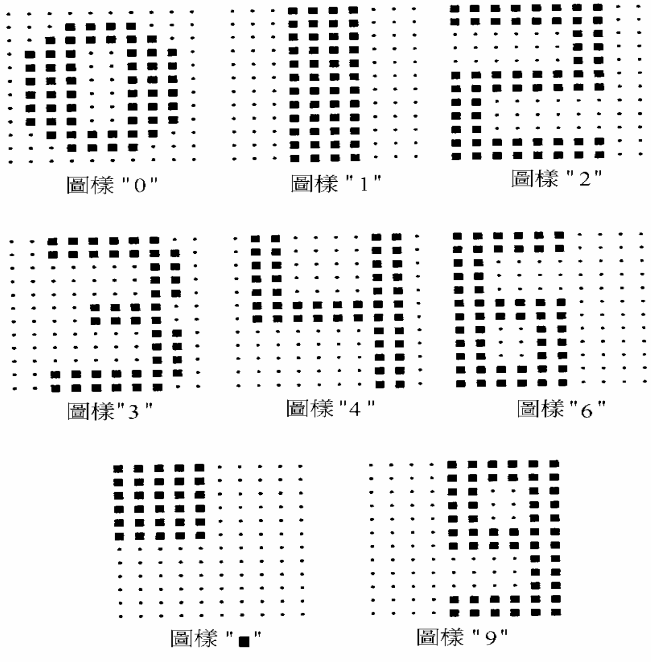
$$v_i^{k+1} = \operatorname{sgn}\left(\sum_{j=1}^{n} w_{ij} v_j^k \right)$$

Test input : $v^0 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$ $v^1 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}$ $v^2 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}$ $v^3 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}$ $v^4 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}$ $v^5 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}$ $v^6 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}$ ---------

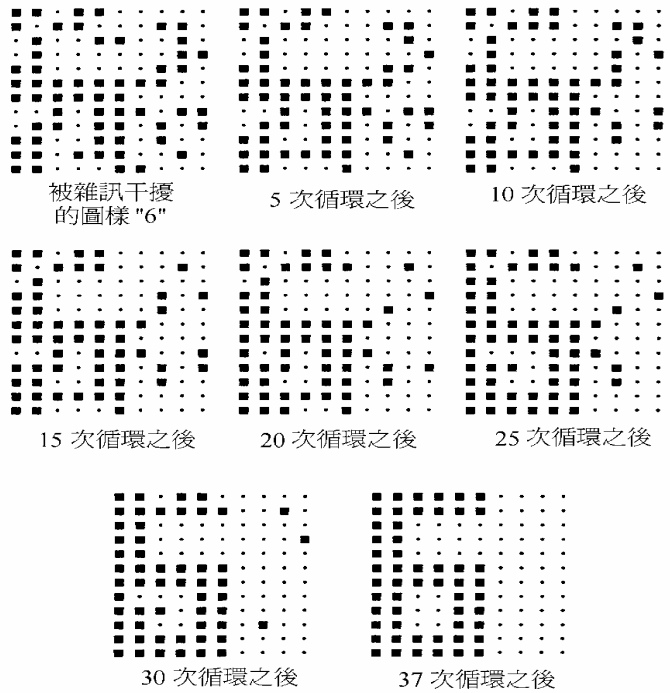
Neuron 1 Neuron 2 Neuron 3 Neuron 1 Neuron 2 Neuron 3

Hopfield's Autoassociative Memory

- **Example**: Autoassociation for numbers (Lippmann 1987)



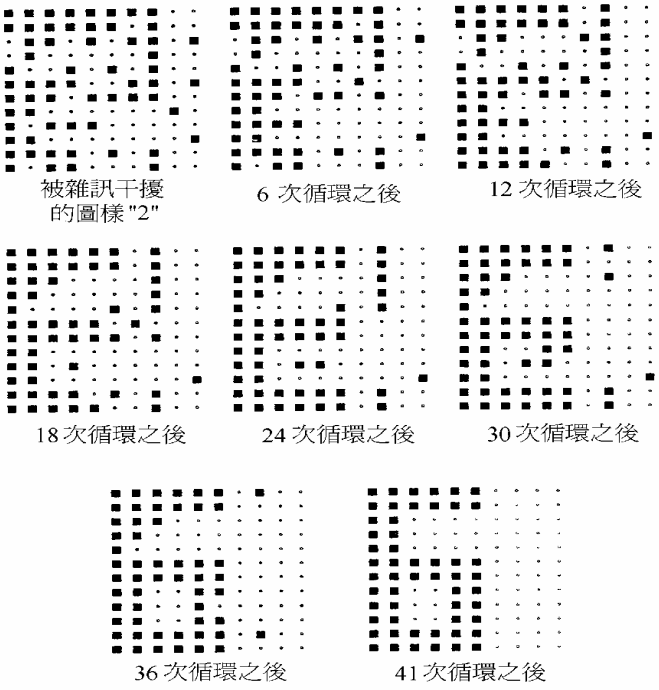
Stored Patterns



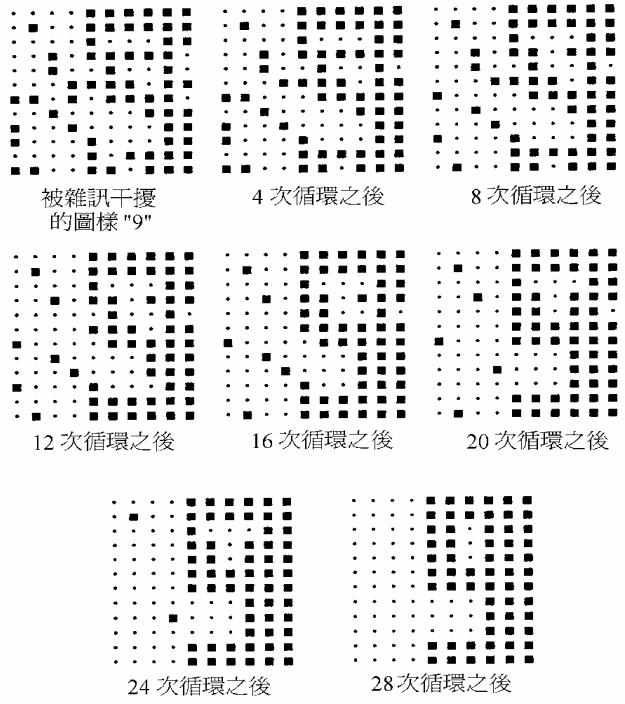
A Distorted Input Pattern "6"

Hopfield's Autoassociative Memory

- **Example**: Autoassociation for numbers



A Distorted Input Pattern "2"



A Distorted Input Pattern "9"

Hopfield's Autoassociative Memory

- IF Unipolar Binary Patterns Used
  - **Scale** and **shift** the association patterns

$$w_{ij} = \left(1 - \delta_{ij}\right) \sum_{m=1}^{p} \left(2s_i^{(m)} - 1\right)\left(2s_j^{(m)} - 1\right)$$

- Performance Consideration
  - Hopfield autoassociative memory also referred to as "Error Correcting Decoder"
  - Given an input equal to the stored memory with random error, it produces as output the original memory that is closed to the input

Hopfield's Autoassociative Memory

- Performance Consideration
  - The *i*-th neuron for distorted retrieval pattern $\boldsymbol{s}^{(m')}$ is considered:

$$net_i = \sum_{j=1}^{n} w_{ij} s_j^{(m')}$$

$$\cong \sum_{j=1}^{n} \sum_{m=1}^{p} s_i^{(m)} s_j^{(m)} s_j^{(m')}$$

If the effect of the diagonal is neglected

$$\cong \sum_{m=1}^{p} s_i^{(m)} \sum_{j=1}^{n} s_j^{(m)} s_j^{(m')}$$

If $\boldsymbol{s}^{(m')}$ and $\boldsymbol{s}^{(m)}$ are statistically independent (also when orthogonal) the product vanished
Otherwise, the product will reach *n*

$$\approx s_i^{(m)} \tilde{n} \quad , \text{where } \tilde{n} \le n$$

Hopfield's Autoassociative Memory

- Performance Consideration
  - The equilibrium state for the retrieval pattern $\boldsymbol{s}^{(m')}$ is considered

$$\boldsymbol{net} = W\boldsymbol{s}^{(m')}$$

$$= \left(\sum_{m=1}^{p} \boldsymbol{s}^{(m)}\boldsymbol{s}^{(m)t} - p\boldsymbol{I} \right)\boldsymbol{s}^{(m')}$$

    - If the stored prototypes are statistically independent or orthogonal, and $n>p$

$$\boldsymbol{net} = (n - p)\boldsymbol{s}^{(m')}$$

The network will be in equilibrium state $s^{(m')}$

    - the stored prototypes are not statistically independent or orthogonal

Noise vector

$$\boldsymbol{net} = (n - p)\boldsymbol{s}^{(m')} + \sum_{m \neq m'}^{p} \left(\boldsymbol{s}^{(m)}\boldsymbol{s}^{(m)t}\right)\boldsymbol{s}^{(m')}$$

$$\boldsymbol{\eta} = \sum_{m \neq m'}^{p} \left(\boldsymbol{s}^{(m)}\boldsymbol{s}^{(m)t}\right)\boldsymbol{s}^{(m')} = \left(W - \boldsymbol{s}^{(m')}\boldsymbol{s}^{(m')t} + \boldsymbol{I}\right)\boldsymbol{s}^{(m')}$$

Hopfield's Autoassociative Memory

- the stored prototypes are not statistically independent or orthogonal
  - Comparison of the equilibrium term in each neuron input
    - » When the *i*-th element of the noise vector is larger than $(n - p)\, s_i^{(m')}$ and has opposite sign, then $s_i^{(m')}$ will not be the network's equilibrium
  - The noise term obviously increases for an increased number of stored patterns, and also become relatively significant when the factor *(n-p)* decreases

Performance Analysis for Recurrent Autoassociation Memory

- **Example 6.2**

$$\boldsymbol{s}^{(1)} = \begin{bmatrix} -1 \\ -1 \\ 1 \\ 1 \end{bmatrix}, \boldsymbol{s}^{(2)} = \begin{bmatrix} -1 \\ 1 \\ 1 \\ -1 \end{bmatrix}$$

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

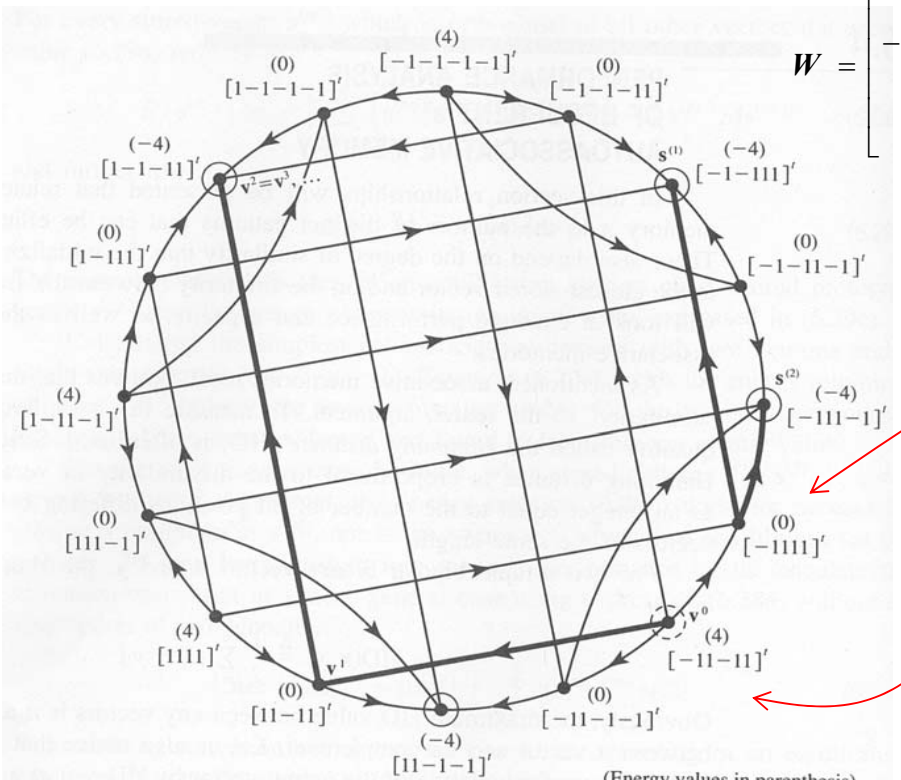


Figure 6.11 Map of transitions for the Example 6.2.

$$\boldsymbol{v}^0 = \begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \end{bmatrix} \qquad \boldsymbol{v}'^0 = \begin{bmatrix} -1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Different update order will lead to different results.

Advantages and Limits for Associative Recurrent Memories

- Advantage
  - The recurrences through the thresholding layer of processing neurons tend to eliminate noise superimposed on the initializing input vector
- Limits
  - Limited capability
  - Converge to spurious memories (states)