Neuralne mreže

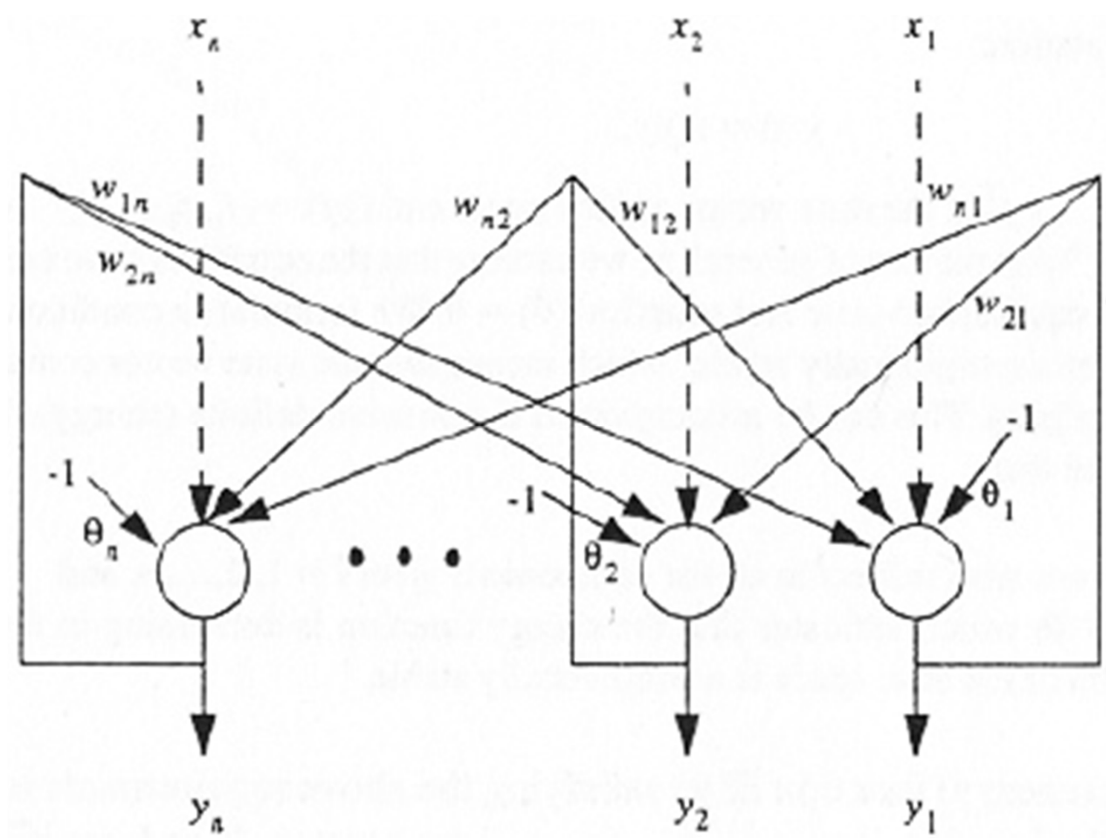
Hopfield networks

Hopfield Networks

- Hopfield's papers [1982,1984] started the modern era in neural networks
- Construction of the first analog VLSI neural chip [1988]
- Single-layer feedback networks with symmetric weights
- Discrete and continuous time

Discrete time Hopfield network *recurrent.* Structure

- Input pattern is first applied to the network, and then removed
- The transition process continues until no new updated responces are produced and network reached its equilibrium



Structure

- Updating rule

$$y_i^{(k+1)} = \text{sgn}\left(\sum_{\substack{j=1 \\ j \neq i}}^{n} w_{ij} y_j^{(k)} + x_i - \theta_i\right), \qquad i = 1, 2, \ldots, n,$$

$$a(f) = \text{sgn}(f) = \begin{cases} 1 & \text{if } f \geq 0 \\ -1 & \text{if } f < 0, \end{cases}$$

- No self feedback $w_{ii} = 0$
- Network weights are symmetric $w_{ij} = w_{ji}$
- Asynchronous update – only one node is updated at one moment

Example

- Two node Hopfield network
- Inputs $x = 0,\ \theta_1 = \theta_2 = 0$
- Weights $w_{12} = w_{21} = -1, w_{11} = w_{22} = 0$
- Initial output vector $y^{(0)} = [-1, -1]^T$
- First node update. Assume now that the first node is chosen for update:

$$y_1^{(1)} = \operatorname{sgn}\left(w_{12} y_2^{(0)}\right) = \operatorname{sgn}[(-1)(-1)] = 1$$

Example

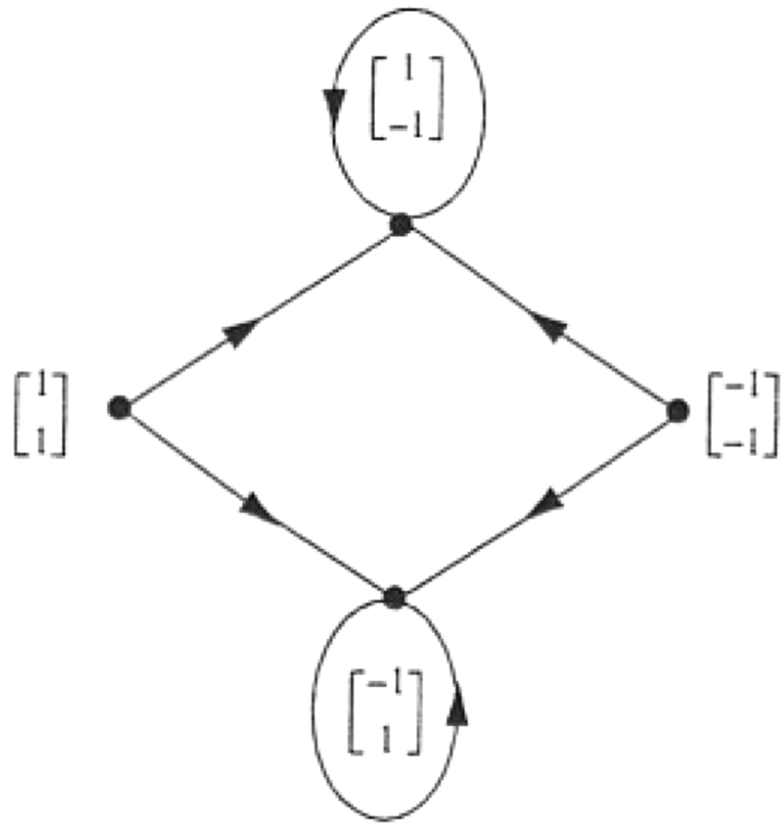
- Now, $y^{(1)} = [1, -1]^T$. Next, the second node is considered for update;

$$y_2^{(2)} = \mathrm{sgn}\left(w_{21} y_1^{(1)}\right) = \mathrm{sgn}[(-1)(1)] = -1$$

- Now, $y^{(2)} = [1, -1]^T$
- It can be easily found that no further output state changes will occur, and $y^{(2)} = [1, -1]^T$ is network equilibrium.

Example

- Using different initial outputs, we can obtain the state transition diagram, in which the vectors $[1, -1]^T$ and $[-1, 1]^T$ are the two equilibria of the system



Example (synchronous update)

- Initial output vector $[-1\ \ -1]^T$

$$\boldsymbol{y}^{(1)} = \begin{bmatrix} \operatorname{sgn}\left[w_{12} y_2^{(0)}\right] \\ \operatorname{sgn}\left[w_{21} y_1^{(0)}\right] \end{bmatrix} = \begin{bmatrix} \operatorname{sgn}[(-1)(-1)] \\ \operatorname{sgn}[(-1)(-1)] \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\boldsymbol{y}^{(2)} = \begin{bmatrix} \operatorname{sgn}\left[w_{12} y_2^{(1)}\right] \\ \operatorname{sgn}\left[w_{21} y_1^{(1)}\right] \end{bmatrix} = \begin{bmatrix} \operatorname{sgn}[(-1)(1)] \\ \operatorname{sgn}[(-1)(1)] \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

- Thus, the result gives back the same vector $\boldsymbol{y}^{(0)}$. Hence, the synchronous update produces a cycle of two states rather than a single equilibrium state.

Stability property of a discrete Hopfield network

- We can characterize the behavior of this network by an energy function *E* as

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1 \\ j\neq i}}^{n} w_{ij}y_i y_j - \sum_{i=1}^{n} x_i y_i + \sum_{i=1}^{n} \theta_i y_i$$

- The idea is to show that if the network is stable, then the above energy function always decreases whenever the state of any node changes.

Stability property of a discrete Hopfield network

- Let us assume that node i has just changed its state from $y_i^{(k)} = y_i^{(k+1)}$. In other words, its output has changed from $+1$ to -1, or vice versa. The change in energy ΔE is then

$$\Delta E = E\left(y_i^{(k+1)}\right) - E\left(y_i^{(k)}\right)$$

$$= -\left(\sum_{\substack{j=1 \\ j\neq i}}^{n} w_{ij} y_j^{(k)} - x_i - \theta_i\right) \left(y_j^{(k+1)} - y_j^{(k)}\right)$$

- Or briefly $\Delta E = -(\mathrm{net}_i)\Delta y_i$, where $\Delta y_i = y_i^{(k+1)} - y_i^{(k)}$

Stability property of a discrete Hopfield network

- If y_i has changed from $y_i^{(k)} = -1$ to $y_i^{(k+1)} = +1$, $(\Delta y_i = 2)$, $\text{net}_i > 0$, ΔE will be negative
- If y_i has changed from $y_i^{(k)} = 1$ to $y_i^{(k+1)} = -1$, $(\Delta y_i = -2)$, $\text{net}_i < 0$, ΔE will be negative
- If y_i has no changed, $\Delta E = 0$
- Finally:

$$\Delta E \leq 0$$

Stability property of a discrete Hopfield network

- Since the energy function E is in quadratic form and is bounded, E must have an absolute minimum value.
- Hence, the energy function, under the update rule, has to reach its minimum (probably a local minimum). Thus, starting at any initial state, a Hopfield network always **converges to a stable state in a finite number of node-updating steps**, where every stable state lies at a **local minimum** of the energy function *E.*

Associative memories

- An associative memory can store a set of patterns as memories. When the associative memory is presented with a *key pattern,* it responds by producing whichever one of the stored patterns most closely resembles or relates to the key pattern.
- Hence, the recall is through association of the key pattern with the information memorized.
- Such memories are also called ***content-addressable*** *memories* in contrast to the traditional *address-addressable memories* in digital computers in which a stored pattern (in bytes) is recalled by its address.
- The basic concept of using Hopfield networks as associative memories is to interpret the system's evolution as a **movement of an input pattern toward the one stored pattern** most resembling the input pattern.

Associative memories

- Two types of associative memories can be distinguished
  - *autoassociative memory* $\Phi\left(\boldsymbol{x}^i\right) = \boldsymbol{x}^i$ $(R^n \to R^n)$
  - If some arbitrary pattern $\boldsymbol{x}$ is *closer* to $\boldsymbol{x}_i$ than to any other $\boldsymbol{x}^j$, $j \neq i$, then $\Phi(\boldsymbol{x}) = \boldsymbol{x}^i$
  - *heteroassociative memory* $\Phi\left(\boldsymbol{x}^i\right) = \boldsymbol{y}^i$ $(R^n \to R^m)$
  - If some arbitrary pattern $\boldsymbol{x}$ is *closer* to $\boldsymbol{x}_i$ than to any other $\boldsymbol{x}^j$, $j \neq i$, then $\Phi(\boldsymbol{x}) = \boldsymbol{y}^i$
- "closer" means with respect to some proper distance measure, for example, the Euclidean distance or the Hamming distance (HD)

Distance

- The *Euclidean distance* d between two vectors

$$d = [(x_1 - x_1')^2 + \cdots + (x_n - x_n')^2]^{\frac{1}{2}}$$

- Hamming distance is defined as the number of mismatched components of x and x' vectors

$$\mathrm{HD}(\mathbf{x}, \mathbf{x}') = \begin{cases} \sum_{i=1}^{n} |x_i - x_i'| & \text{if } x_i, x_i' \in \{0, 1\} \\ \frac{1}{2}\sum_{i=1}^{n} |x_i - x_i'| & \text{if } x_i, x_i' \in \{-1, 1\}. \end{cases}$$

Linear associator

- In a special case where the vectors x_i, $i = 1, 2, \ldots, p$, form an orthonormal set, the associative memory can be defined as
  $\Phi(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{x} = (y^1(x^1)^T + y^2(x^2)^T + \cdots + y^p(x^p)^T)\boldsymbol{x}$
- where $\boldsymbol{W}$ can be considered a weight matrix, called a *cross-correlation* matrix, of the network.

Recurrent Autoassociative Memory
Hopfield Memory

- A Hopfield memory is able to recover an original stored vector when presented with a probe vector close to it.
- In Hopfield memory, data *retrieval rule* that is applied asynchronously and stochastically
- The remaining problem is how to store data in memory. Assume **bipolar binary vectors** that need to be stored are $\boldsymbol{x}^k$ for $k = 1, 2, \ldots, p$. The *storage algorithm* for finding the weight matrix is

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

Recurrent Autoassociative Memory
Hopfield Memory

- Equivalent form

$$w_{ij} = \sum_{k=1}^{p} x_i^k x_j^k \qquad i \neq j\,; w_{ii} = 0$$

- A Hopfield memory is able to recover an original stored vector when presented with a probe vector close to it.
- In Hopfield memory, data *retrieval rule* that is applied asynchronously and stochastically
- The remaining problem is how to store data in memory. Assume bipolar binary vectors that need to be stored are $\boldsymbol{x}^k$ for $k = 1, 2, \ldots, p$. The *storage algorithm* for finding the weight matrix is

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

Recurrent Autoassociative Memory
Hopfield Memory

- If x^i are unipolar binary vectors, that is, $x \in \{0, 1\}$, then the storage rule is

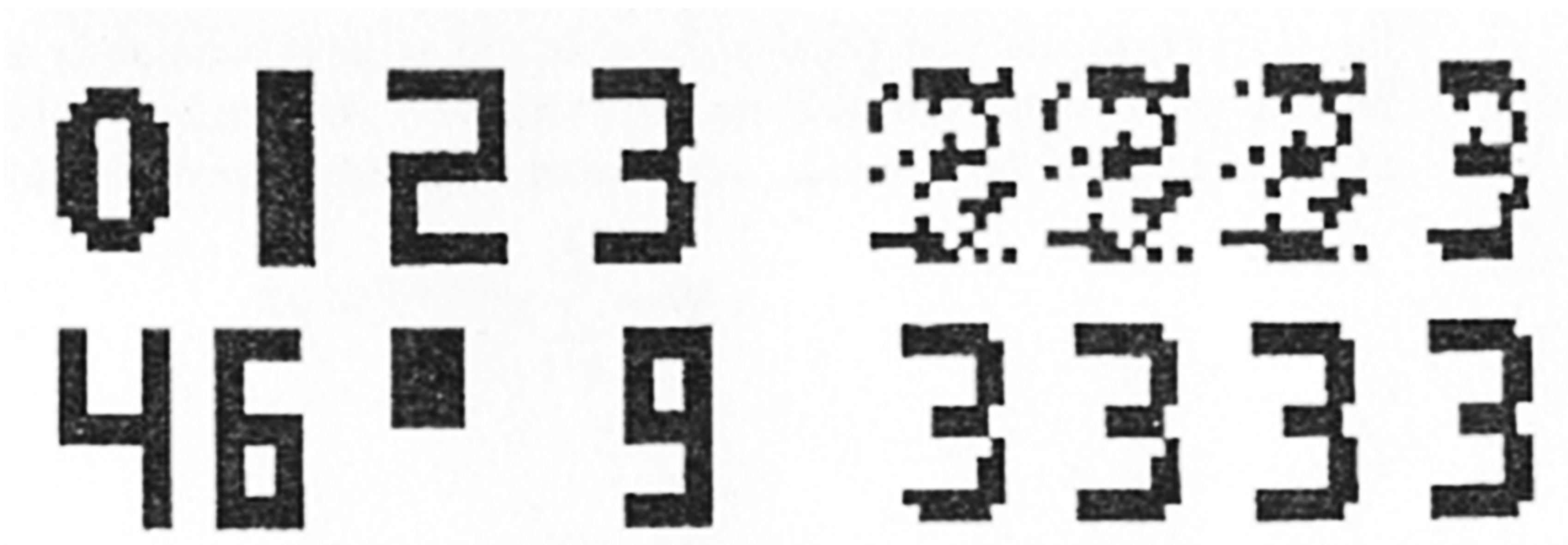
$$w_{ij} = \sum_{k}^{p} (2x_i^k - 1)(2x_j^k - 1), \qquad i \neq j\,; w_{ii} = 0$$

Example

- A Hopfield memory with 120 nodes and thus 14,400 weights is used to store the eight examplar patterns.
- Input elements to the network take on the value + 1 for black pixels and -1 for white pixels.
- In a test of recalling capability, the pattern for the digit 3 is corrupted by randomly reversing each bit independently from + 1 to -1, and vice versa, with a probability of 0.25.
- This corrupted pattern is then used as a key pattern and applied to the Hopfield network at time zero.

Example

- The states of the network for iterations 0 to 7 are shown.
- It is clear that the network converges to the digit 3 pattern correctly.



Example

- Consider the use of a Hopfield memory to store the two vectors $\boldsymbol{x}^1$ and $\boldsymbol{x}^2$

$$\boldsymbol{x}^1 = [1\ -1\ -1\ \ 1]^T;\ \boldsymbol{x}^2 = [-1\ 1\ -1\ \ 1]^T;$$

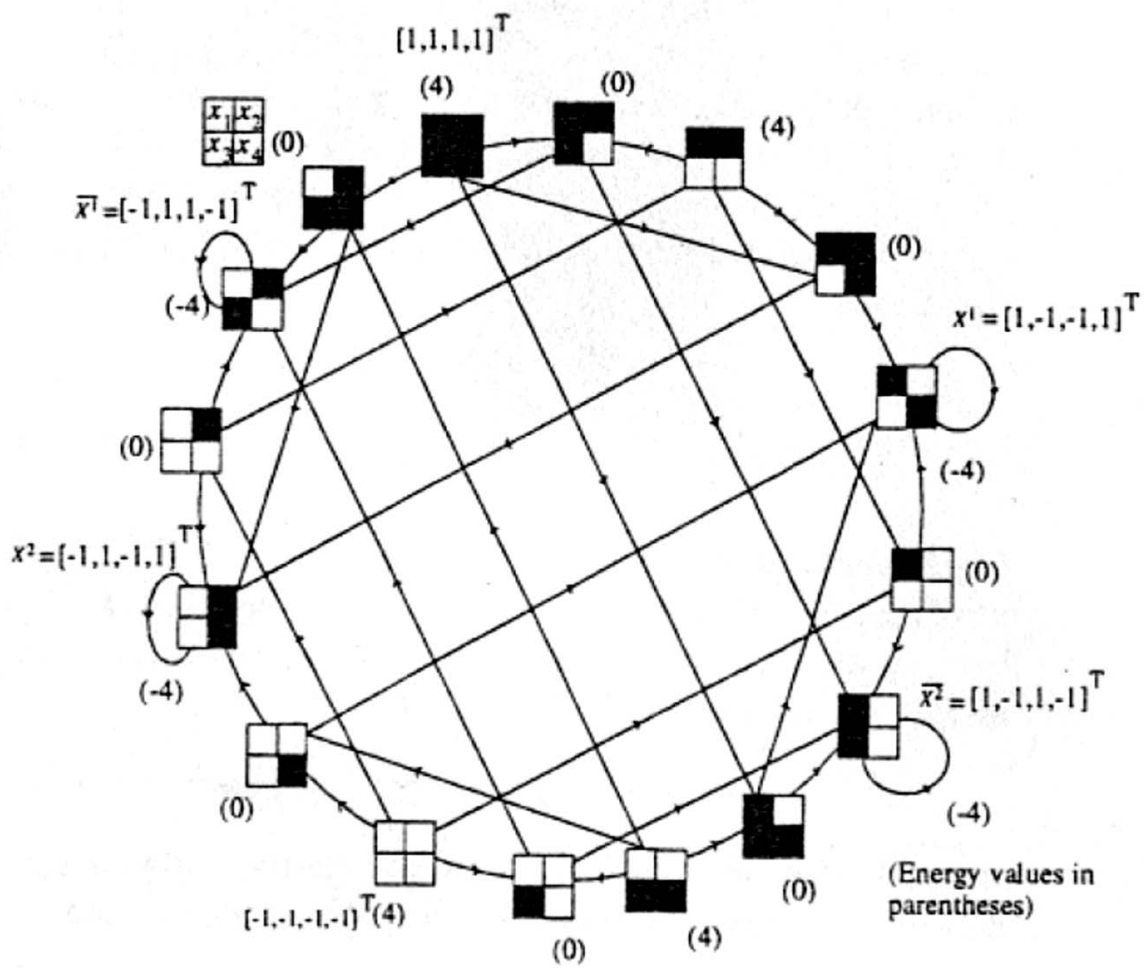
- We obtain the weight matrix as

$$\boldsymbol{W} = \sum_{k=1}^{2} x^k \left(x^k\right)^T - 2\boldsymbol{I} = \begin{bmatrix} 0 & -2 & 0 & 0 \\ -2 & 0 & 0 & 0 \\ 0 & 0 & 0 & -2 \\ 0 & 0 & -2 & 0 \end{bmatrix}$$

- The energy function is

$$E(\boldsymbol{x}) = -\frac{1}{2}\boldsymbol{x}^T\boldsymbol{W}\boldsymbol{x} = 2(x_1x_2 + x_3x_4)$$

The state transition diagram



Example

- There are a total of 16 states, each of which corresponds to *one vertex.*
- Figure shows all possible asynchronous transitions and their directions. Note that every vertex is connected only to the neighboring vertex differing by a single bit because of asynchronous transitions.
- Each state is associated with its energy value. **It is observed that transitions are toward lower energy values.**
- There are two extra stable states $\bar{x}^1 = [-1\ 1\ \ 1\ -1]^T$ and $\bar{x}^2 = [1\ -1\ \ 1\ -1]^T$

Transition examples

- Starting at the state $[1, 1, 1, 1]$ and with nodes updating asynchronously in ascending order, we have state transitions
- $[1, 1, 1, 1] \rightarrow [-1,1,1,1] \rightarrow [-1,1,1,1] \rightarrow [-1,1,-1,1]$ ...
- The state will converge at the stored pattern x^2
- However, it is possible (with a different updating order) that the state $[1, 1, 1, 1]$ will converge to x^1, x^2, $\bar{x}^1, \bar{x}^2$
- This happens because the Hamming distance between the initial state, $[1, 1, 1, 1]$ and any of x^1, x^2, $\bar{x}^1, \bar{x}^2$ is of the same value 2.

Important fact

- The above example indicates an important fact about the Hopfield memory - that the ***complement*** **of a stored vector is also a stored vector.**
- The reason is that they have the same energy value $E(\boldsymbol{x}) = E(\overline{\boldsymbol{x}})$.
- Hence, the memory of transitions may terminate as easily at $\boldsymbol{x}$ as at $\overline{\boldsymbol{x}}$.
- The crucial factor determining the convergence is the "similarity" between the initializing output vector and $\boldsymbol{x}$ and $\overline{\boldsymbol{x}}$.

Problems of Hopfield memories

- Two major problems of Hopfield memories are observed from the above example
  - The first is the unplanned stable states, called *spurious stable states,* which are caused by the minima of the energy function in addition to the ones we want.
  - The second is uncertain recovery, which concerns the *capacity* of a Hopfield memory. Overloaded memory may result in a small Hamming distance between stored patterns and hence does not provide error-free or efficient recovery of stored patterns.

Problems of Hopfield memories

- It has been observed that the relative, number of spurious states decreases as the dimensionality of the stored vectors (i.e. the number of neurons n) increases with respect to the number of stored vectors.
- Eventually, a point is reached where there are relatively so few within a certain Hamming radius of each original stored vector that it becomes valid to consider each memory as having a fixed; radius of convergence.