Associative Memories
Hopfield Network

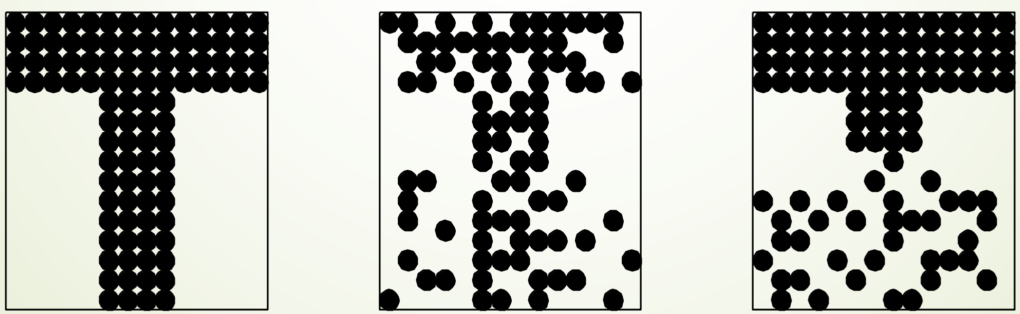
(Revised slides from HOU-PLH31)

I. Hatzilygeroudis

Dept of Computer Engineering & Informatics, University of Patras

Associative Memory

- Recall of an event at a point in time is caused by the association of that event with some stimulus.
- Many times, we are also asked to recognize partially damaged letters or to recognize through a window while it is raining (in the presence of noise).

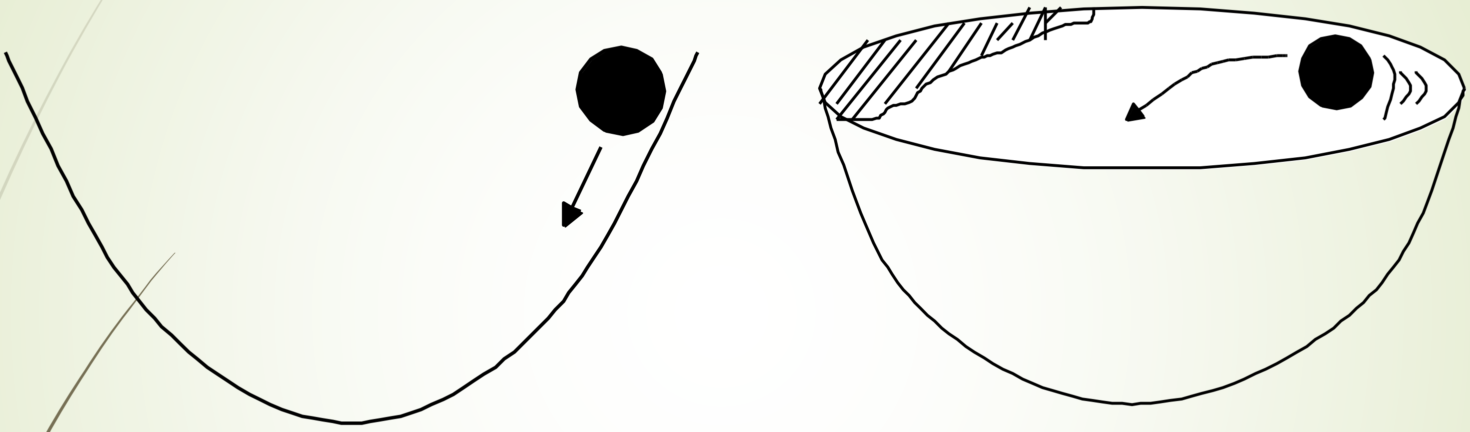


ORIGINAL T

20% NOISE IN WHOLE IMAGE

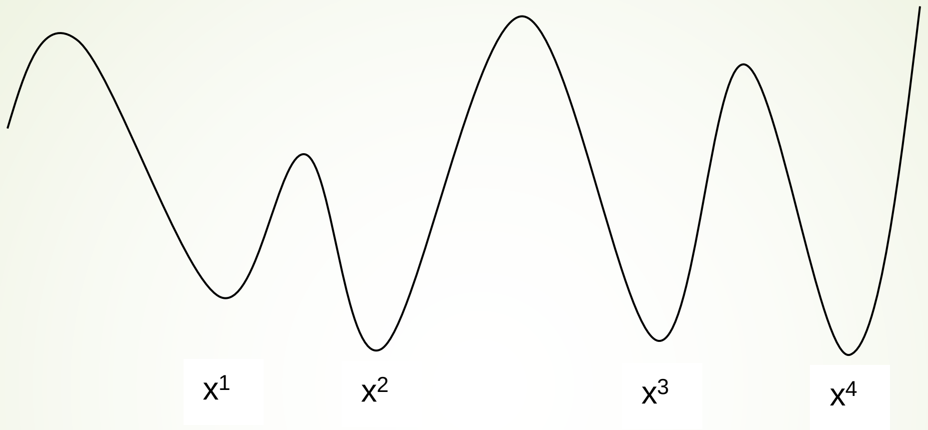
HALF IMAGE DESTROYED BY NOISE

Mechanical analog of associative memory



- Start position: **stimulus**, end position: **recall**
- The ball always ends up at the same point (**equilibrium point**) because it **'remembers'** where the bottom of the container is (**state of minimum potential energy**).

Mechanical analog of associative memory



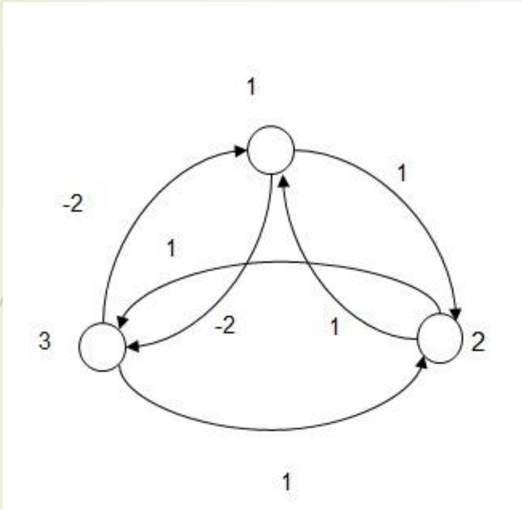
x^1, x^2, x^3, x^4 are the stored states

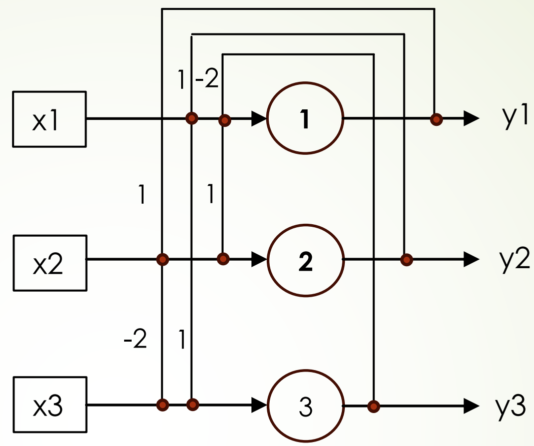
- If the sphere starts from some random point, then it will end up at the nearest local concavity (local minimum).
- Therefore, it recalls the nearest stored pattern. This point is also a **local minimum of the energy** of the system.

Associative memory system

- The state of the system is described by a **state vector** x=(x1,x2,…,xn).
- There exists a set of **equilibrium states** {x1,x2,…,xm}, where xi=(xi1,xi2,…,xin). These correspond to the stored examples and are the local minima of the energy of the system.
- The system starts from an **initial state (stimulus)** and ends up in one of the **equilibrium states (recall**) corresponding to one of the stored patterns, also called **basic memories**. This process is accompanied by a **reduction in the energy E** of the system.

Hopfield Network





Input
x= (x_1, x_2, x_3)

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

Output
y= (y_1, y_2, y_3)

- Fully connected
- Recursive connections
- **Symetric weights ($w_{ij} = w_{ji}$)**, w_{ii}=0.
- Biases b_i.
- **Recurrent ANN**.

Hopfield Network

- **Discrete-time and discrete-output** Hopfield network:
  - time progresses in discrete steps t, t+1,…
  - neuron outputs are discrete: bipolar values (**1 or -1**) or binary values (**1 or 0**).
- In such a network with n neurons examples can be stored such as

$$x= (x1,\dots,xn) \text{ (where xi is e.g. -1 or 1).}$$

- We want these examples to be equilibrium situations of the network, with an appropriate selection of the weights w_{ij}.

Hopfield Network

- **Update of the status (output) of a node** i at time point t:

$$u_i(t)=\sum_{j=1}^{n} w_{ij}\, y_j(t)+b_i \longrightarrow y_i(t+1)=\mathrm{sgn}(u_i(t))$$

$\mathrm{sgn}(u)=1$ if $u>0$, $\mathrm{sgn}(u)=-1$ if $u<0$.

if $u_i(t)=0$, θέτουμε $y_i(t+1)=y_i(t)$

- Definition: A situation $y(t)=(y_1(t),\ldots,y_n(t))$ is an **equilibrium situation** of a Hopfield network, when $y_i(t+1)=y_i(t)$ for each $i=1,\ldots,n$.
- A situation $y=(y_1,\ldots,y_n)$ is an equilibrium situation , if and on;y if satisfies the **equilibrium condition** for each $i=1,\ldots,n$:

$$y_i=\mathrm{sgn}\Big(\sum_{j=1}^{n} w_{ij}\, y_j + b_i\Big) \Leftrightarrow y_i\cdot\Big(\sum_{j=1}^{n} w_{ij}\, y_j + b_i\Big) \geq 0$$

Synchronous operation of Hopfield network

- Initialization: t=0, application of the input example x and specification of the initial state y(0) by setting yi(0)=xi
- At each time t, let y(t)=(y1(t),…,yn(t))T be the state of the network. All its nodes are **simultaneously** updated by first computing ui(t) for all i:

$$u_i(t)= \sum_{j=1}^{n} w_{ij}\ y_j(t) + b_i$$

  and in the sequel the $y_i(t+1)=sgn(u_i(t))$ for all i.
- It is shown that the network will **either reach an equilibrium state or engage in a cycle of length two**, i.e. it will continuously oscillate between two states.

Asynchronous operation of Hopfield network

- Initialization: t=0, application of the input example x and specification of the initial state y(0) by setting yi(0)=xi
- At each time t, let $y(t)=(y1(t),\ldots,yn(t))^T$ be the state of the network.
  - ✓ Selection of a neuron i.
  - ✓ Update output $y_i(t+1)$:

$$u_i(t) = \sum_{j=1}^{n} w_{ij}\, y_j(t) + b_i \qquad y_i(t+1) = \mathrm{sgn}(u_i(t))$$

  - ✓ t:=t+1
- Until an equilibrium situation/state is reached: $y_i(t+1)=y_i(t)$ for each $i=1,\ldots,n$.
- The final equilibrium situation/state constitutes the response of the network to the stimulus y(0)=x.

Asynchronous operation of Hopfield network

- In practice the neurons are selected for updating in order rather than randomly. When all neurons have been examined once, we consider an epoch to be completed. The update order can be changed every epoch or kept the same.
- We consider that we have reached a state of equilibrium when an epoch is completed (every node has been examined) and the output of no neuron has changed.
- **Asynchronous operation guarantees the convergence** of the network to equilibrium because the network is characterized by a function called the **energy function** :

$$E(t) = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i(t)\, y_j(t)\, w_{ij} - \sum_{i=1}^{n} b_i\, y_i(t)$$

Example

- A Hopfield network of two neurons is given, with weights values $w_{12}=w_{21}=-1$ and zero biases.
- Let the initial status (t=0) is y(0)=(1,1), i.e. $y_1(0)=1$ and $y_2(0)=1$.
- If we apply **synchronous update,** we observe that:

$$y_1(t=1)=sgn(w_{21}y_2(0)+b_1)=sgn(-1)=-1$$

$$y_2(t=1)=sgn(w_{12}y_1(0)+b_2)=sgn(-1)=-1$$

- Hence y(1)=(-1,-1). Next, we calculate for t=2:

$$y_1(t=2)=sgn(w_{21}y_2(1)+b_1)=sgn(1)=1$$

$$y_2(t=1)=sgn(w_{12}y_1(1)+b_2)=sgn(1)=1$$

- Consequently y(2)=(1,1)=y(0) (cycle of length 2).
- The same we find for the initial state y=(-1,-1).

Example

- Let us now consider the case where the initial state is y(0)=(1,-1) and the network again operates with **synchronous updating**.
  - $y_1(t=1)=sgn(w_{21}y_2(0)+b_1)=sgn(1)=1$
  - $y_2(t=1)=sgn(w_{12}y_1(0)+b_2)=sgn(-1)=-1$
- So, y(1)=(1,-1)=y(0), δηλ. η y=(1,-1) is an equilibrium situation.
- The same we find for the initial state y=(-1,1).

Example

- **Asynchronous update** with initial state y(0)=(1,1). Suppose that neuron 1 is selected first for updating :

$$y_1(t=1)=\mathrm{sgn}(w_{21}y_2(0)+b_1)=\mathrm{sgn}(-1)=-1$$

$$y_2(t=1)=1 \text{ (not updated)}$$

- So, y(1)=(-1,1). Next, neuron 2 is selected:

$$y_1(t=2)=-1 \text{ (not updated)}$$

$$y_2(t=2)=\mathrm{sgn}(w_{12}y_1(1)+b_2)=\mathrm{sgn}(1)=1$$

- So, y(2)=(-1,1). Then if we examine neuron 1 again, we will see that its state will not change, so we are in **equilibrium state**.
- .

Design-Operation of Hopfield Net

Determination of architecture and weights

- Suppose we have **M** examples (of **dimension n**) to store as equilibrium states or **basic memories** of a Hopfield network. Let xpi, xpj (with values 1 or -1) be the elements i and j of example p.
- Due to the dimensionality of the examples, the network will have n nodes (neurons).
- Weights are calculated using **Hebb's Rule :**

$$w_{ij} = \sum_{p=1}^{M} x_{pi}\, x_{pj}\ ,\ w_{ii} = 0,\ b_i = 0,\ i,j = 1,\ldots,n$$

that in the matrix form is written: $\mathbf{W} = \sum_{\mathbf{m=1}}^{\mathbf{M}} \mathbf{Y_m} \times \mathbf{Y_m^T} - \mathbf{M} \times \mathbf{I}$

where Ym are the basic memories and I is the identity matrix of dimensions nxn.

Design-Operation of Hopfield Net

- The weight matrix is **symmetric** ($w_{ij} = w_{ji}$) with zero values on the main diagonal ($w_{ii} = 0$). The weights are **calculated once and remain constant**.

$$W = \begin{bmatrix} 0 & w_{12} & \cdots & w_{1i} & \cdots & w_{1n} \\ w_{21} & 0 & \cdots & w_{2i} & \cdots & w_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ w_{i1} & w_{i2} & \cdots & 0 & \cdots & w_{in} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ w_{n1} & w_{n2} & \cdots & w_{ni} & \cdots & 0 \end{bmatrix}$$

- It does not guarantee that the memories will be saved. Also, other patterns may be stored as memories without our desire.

Design-Operation of Hopfield Net

Network testing

- A check is made to store the equilibrium states for each :

$$\mathbf{X_m} = \mathbf{Y_m},\ \text{m}=1,2,\ \ldots\ ,\text{M}$$

$$\mathbf{Y_m} = \mathbf{sign}\left(\mathbf{W} \times \mathbf{X_m} - \boldsymbol{\theta}\right)$$

Normally all basic memories (equilibrium states) should be recalled.

- The **capacity** of a Hopfield network, i.e. the maximum number of equilibrium states (basic memories) it can store, is approximately: C = n/(2log2n), where n is the number of neurons. On the other hand, a more simplified version, C = 1.38*n is also considered.

Design-Operation of Hopfield Net

Network operation

- A "decayed" vector of a basic memory (equilibrium state) is given as input and the system "recalls" the corresponding basic memory.
- For example, a vector representing a "distorted" image is given as input, and the network returns to the output the normal image, already stored in the network as basic memory.

Hopfield example

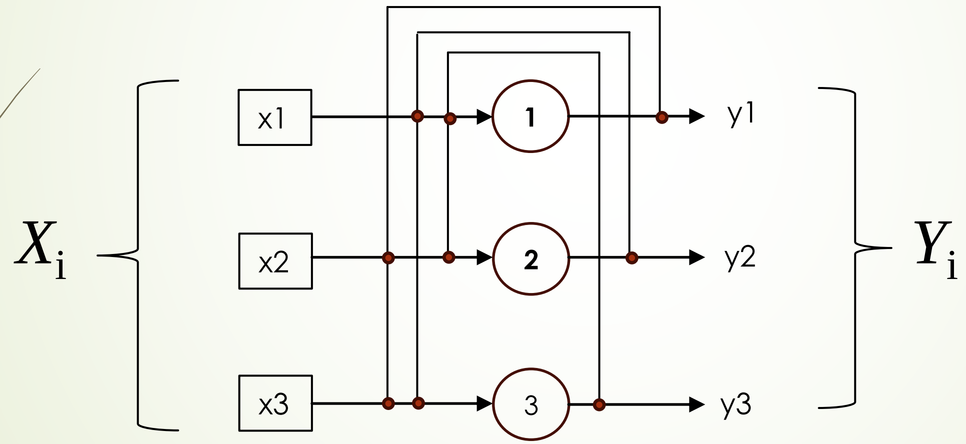
- **Problem:** Design a Hopfield network that stores the following states (basic memories):

$$X_1 = \begin{bmatrix} +1 \\ +1 \\ +1 \end{bmatrix} \quad X_2 = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$$

Assume that the biases are zero and we have synchronous operation.

Παράδειγμα Hopfield

- **Architecture:** Since the vectors have three component values, so the Hopfield network will have three neurons :



Hopfield example

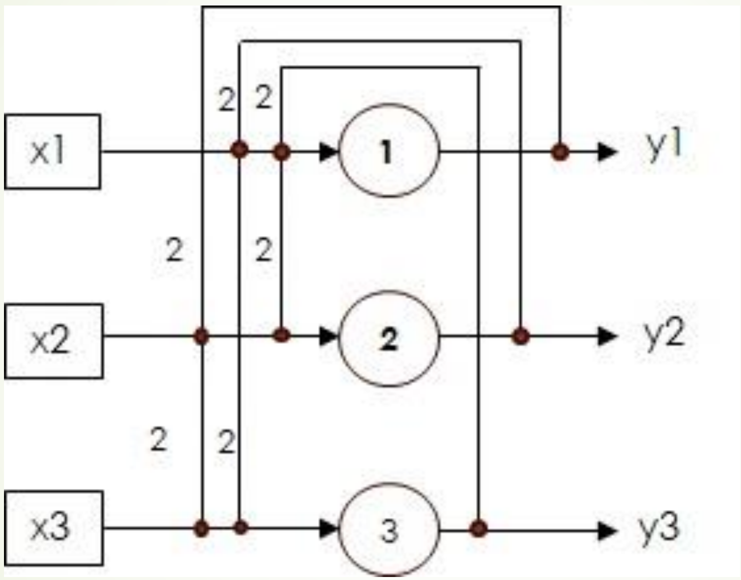
- Calculation of weights matrix:

$$W = \sum_{m=1}^{2} X_m \times X_m^T - M \times I = X_1 \times X_1^T + X_2 \times X_2^T - 2 \times I =>$$

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

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

Hopfield example



Hopfield example

Testing: We test whether the two states have been stored.

We put as input the **first vector**:

$$X_1 = \begin{bmatrix} +1 \\ +1 \\ +1 \end{bmatrix}$$

We calculate the output:

$$Y_1 = sign(W \times X_1 + W_0) = sign\left(\begin{bmatrix} 0 & 2 & 2 \\ 2 & 0 & 2 \\ 2 & 2 & 0 \end{bmatrix}\begin{bmatrix} +1 \\ +1 \\ +1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}\right) =$$

$$= sign\left(\begin{bmatrix} +4 \\ +4 \\ +4 \end{bmatrix}\right) = \begin{bmatrix} +1 \\ +1 \\ +1 \end{bmatrix} = X_1$$

The memory is recalled, so it is stored correctly.

Hopfield example

We put as input the **second vector** :

$$X_2 = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$$

We calculate the output :

$$Y_2 = sign(W \times X_2 + W_0) = sign\left(\begin{bmatrix} 0 & 2 & 2 \\ 2 & 0 & 2 \\ 2 & 2 & 0 \end{bmatrix}\begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}\right) =$$

$$= sign\left(\begin{bmatrix} -4 \\ -4 \\ -4 \end{bmatrix}\right) = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} = X_2$$

The memory is recalled, so it is stored correctly.

Hopfield example

Let's try to insert a "decayed" vector of one of the two main memories (let's say the first one):

$$X_3 = \begin{bmatrix} -1 \\ +1 \\ +1 \end{bmatrix}$$

We calculate the output :

$$Y_3 = sign(W \times X_3 + W_0) = sign\left(\begin{bmatrix} 0 & 2 & 2 \\ 2 & 0 & 2 \\ 2 & 2 & 0 \end{bmatrix}\begin{bmatrix} -1 \\ +1 \\ +1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}\right) =$$

Observe that sign(0) = +1, because when the input x = 0, the output is the same as the previous one.

$$= sign\left(\begin{bmatrix} +4 \\ 0 \\ 0 \end{bmatrix}\right) = \begin{bmatrix} +1 \\ +1 \\ +1 \end{bmatrix} = X_1$$

Basic memory is recalled, so the network operates correctly.