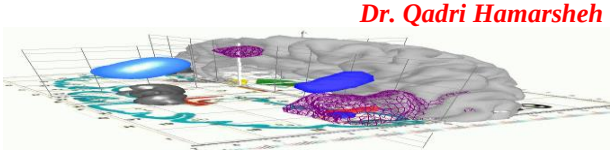
Dr. Qadri Hamarsheh



Neural Networks and Fuzzy Logic (630514)

Lecture 12

Supervised Learning in Neural Networks (Part 5)
Hopfield Networks

There are at least two ways in which we might carry out the updating specified by the equation used in Hopfield Networks:

- ***Synchronously or parallel:*** update all the units simultaneously at each time step.
- ***Asynchronously or Sequential:*** Update them one at a time.

Problems of Hopfield Network

1) **Incorrect convergence (recalling)**: The Hopfield network converges to a stable state if the retrieval is done asynchronously; this stable is not necessarily representing fundamental memory or the closest one.

Example: Store 3 fundamental memories

$$X1 = [1 \quad 1 \quad 1 \quad 1 \quad 1];$$
$$X2 = [1 \quad -1 \quad 1 \quad -1 \quad 1];$$
$$X3 = [-1 \quad 1 \quad -1 \quad 1 \quad -1];$$

The weight matrix

$$W = \begin{bmatrix} 0 & -1 & 3 & -1 & 3 \\ -1 & 0 & -1 & 3 & -1 \\ 3 & -1 & 0 & -1 & 3 \\ -1 & 3 & -1 & 0 & -1 \\ 3 & -1 & 3 & -1 & 0 \end{bmatrix}$$

Probe vector:

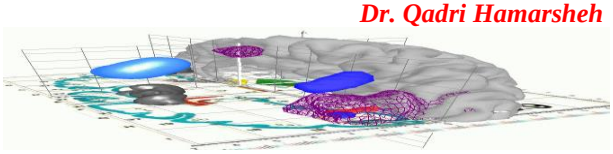
$$X = [1 \quad 1 \quad -1 \quad 1 \quad 1];$$

The output of the network will converge to the fundamental memory $X3$ which is wrong (the correct answer is $X1$)

2) **Storage capacity of the Hopfield network**: the maximum number of patterns (fundamental memories) that can be stored and retrieved correctly without unacceptable errors.

- The maximum number of fundamental memories M_{max} that can be stored in the n-neuron recurrent network is limited by
  - ***a)*** **Hopfield experimentally:** $M_{max} = 0.15\, n$.
  - ***b)*** **Most perfectly retrieved:** $M_{max} = \frac{n}{2 \ln n}$.
  - ***c)*** **All perfectly retrieved:** $M_{max} = \frac{n}{4 \ln n}$

3) **Hopfield network represent an auto-associative type of memory:** The Hopfield network is a single layer and it can store in this layer (the same neurons) the output and the input patterns. To associate one memory with another we need a recurrent network with two layers (one set of neurons for input patterns and another set of neurons for the output patterns) ⇒**Bidirectional Associative Memory**.



Hopfield network training algorithm:

1) **Storage (learning):** In the learning step for Hopfield network we need to find weight matrix for M of patterns (fundamental memories: $Y_1, Y_2, Y_3, \dots, Y_M$,) stored in the synaptic weights of the network according to the equation

$$w_{ij} = \begin{cases} \sum_{m=1}^{M} y_{m,i} y_{m,j}, & i \neq j \\ 0 & i = j \end{cases}$$

$y_{m,i}$ and $y_{m,j}$, ith and the jth elements of the fundamental memories Y_m,

In **matrix form**:

$$W = \sum_{m=1}^{M} Y_m Y_m^T - MI$$

The weight matrix is symmetrical with zeros in the main diagonal

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

Where $w_{ij} = w_{ji}$, The weight matrix **W** is remains fixed

2) **Testing**

- We need to confirm that the Hopfield network is able to recall all fundamental memories:
  - **Recall** Y_m when presented $Y_m$as an input.

Using:

$$x_{m,i} = y_{m,i}, \quad i = 1, 2, \dots, n; \quad m = 1, 2, \dots, M$$
$$y_{m,i} = sign\left(\sum_{j=1}^{n} w_{ij} x_{m,j} - \theta_i\right)$$

Where:

$y_{m,i}$ is the **ith** element of the actual output vector Y_m.

$x_{m,j}$ is the **jth** element of the input vector X_m.

In **matrix form**:

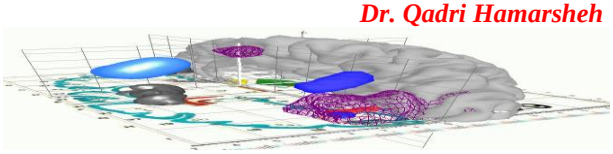
$$X_m = Y_m, \quad m = 1, 2, \dots, M$$
$$Y_m = sign\,(WX_m - \theta)$$

If all fundamental memories are recalled perfectly, then go to the next step

3) **Retrieval**

- Present an unknown n-dimensional vector (**probe**), X, (corrupted or incomplete version of a pattern from fundamental memories) to the network and retrieve a **stored association** (**stable state**):

$$X \neq Y_m, \quad m = 1, 2, \dots, M$$



- **Initialize** the network:

$$x_j(0) = x_j, \qquad j = 1, 2, \dots, n$$

Where $x_j(0)$ is the ***jth*** element of the probe vector X at iteration $p = 0$

In **matrix form**:

$$X(0) = X, \quad p = 0$$

- **Calculate** the network output at iteration $p = 0$:

$$y_i(0) = sign\left(\sum_{j=1}^{n} w_{ij} x_j(0) - \theta_i\right), \qquad i = 1, 2, \dots, n$$

Where $y_i(0)$ is the state of neuron i at iteration $p = 0$

In **matrix form**:

$$Y(0) = sign[WX(0)]$$

- **Update** the elements of state vector $Y(p)$ using the rule

$$y_i(p+1) = sign\left(\sum_{j=1}^{n} w_{ij}\, x_j(p) - \theta_i\right)$$

In **matrix form**:

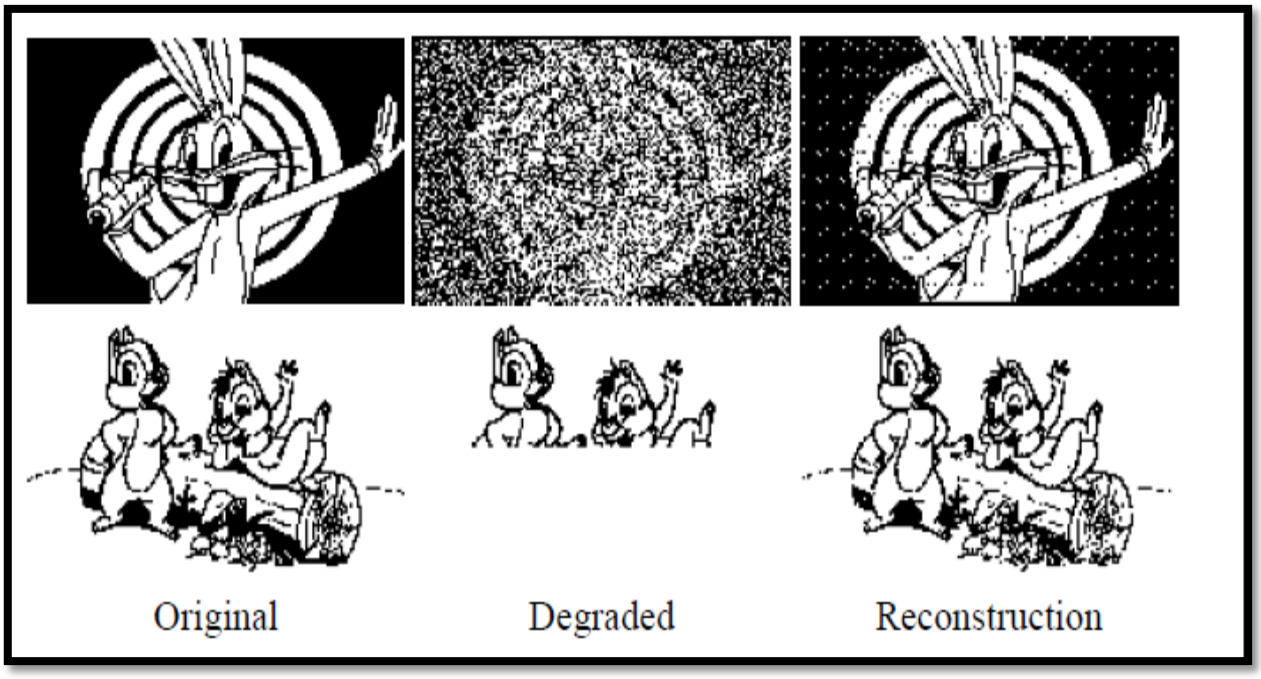
$$Y(p+1) = sign[WX(p) - \theta]$$

- **Repeat** the iteration until convergence, when input and output remain **unchanged**. The **condition for stability** can be defined as :

$$y_i(p+1) = sign\left(\sum_{j=1}^{n} w_{ij}\, y_j(p) - \theta_i\right), \qquad i = 1, 2, \dots, n$$

In **matrix form**:

$$Y(p+1) = sign[WY(p) - \theta]$$



Hopfield network reconstructing degraded images from noisy (top) or partial (bottom)