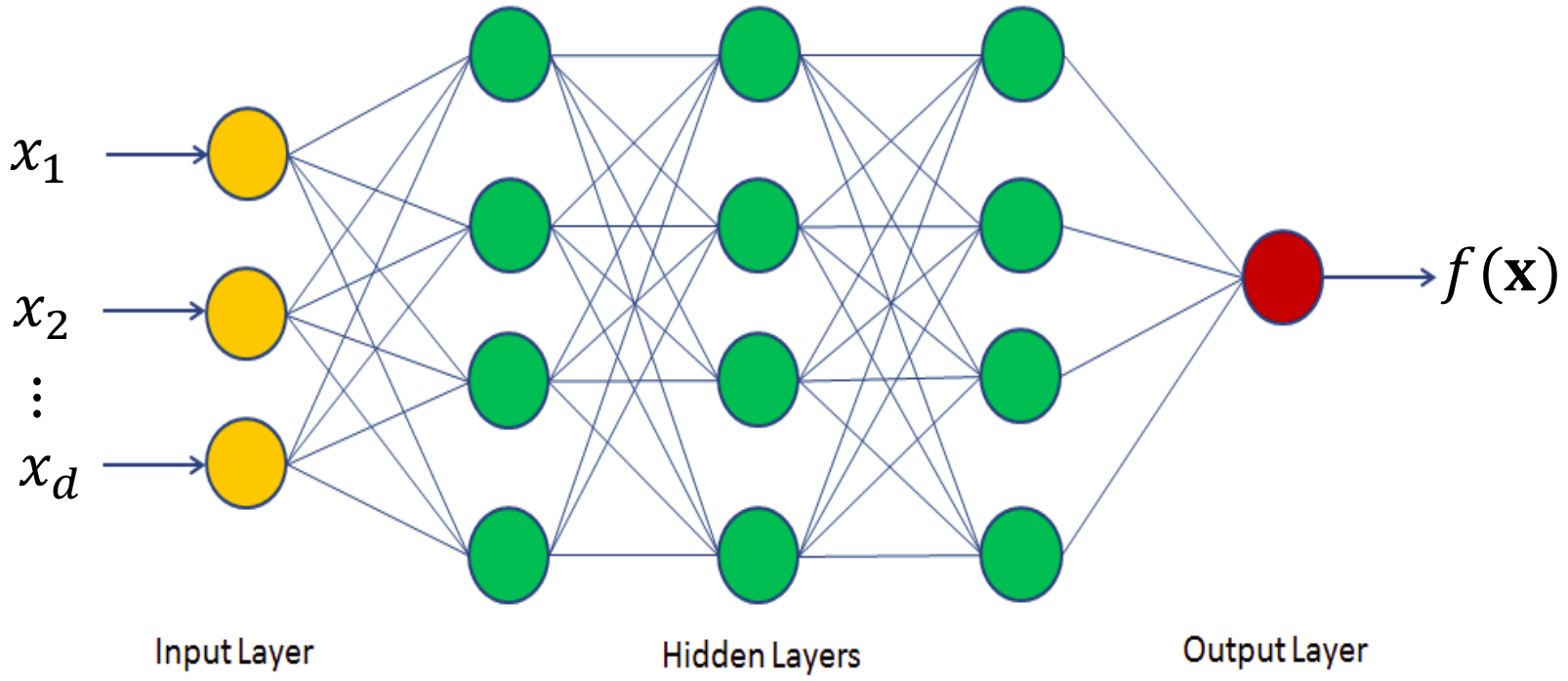
2. Feed-Forward Multi-layer Neural Networks

The architecture of a typical feed-forward multi-layer neural network is shown below:

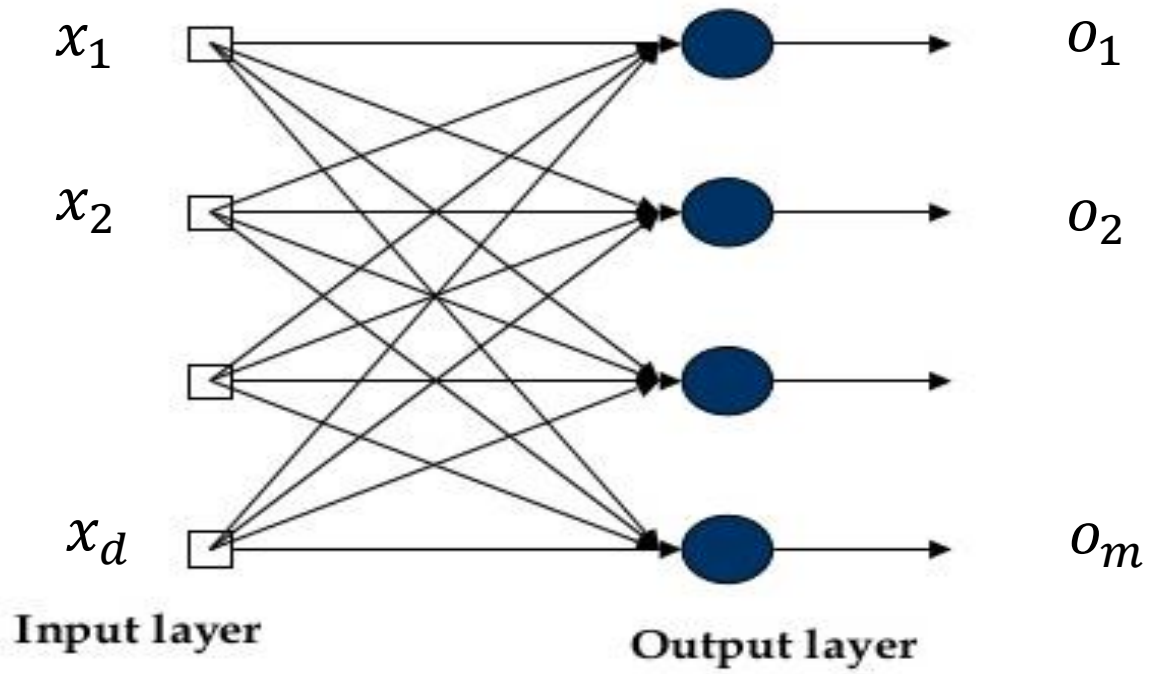


- **Definitions**

  - Input layer: the layer where the input is applied.
  - Output layer: the layer from which the output responses are obtained
  - Hidden layer: the intermediate layers between the output and the input layers.
  - Neurons: represented by circles, the basic units that perform computations
  - Connection and communication path: represented by arrows. In the forward neural networks, the communications are from one layer to the next, without leapfrog layers

- **Single-layer feed-forward network**

The architecture of a single-layer feed-forward neural network is shown below:



The network has *d* input variables and *m* outputs. The input and output vectors are denoted as follows:

$$\mathbf{x} = [x_1, x_2, \cdots, x_d]^T$$

$$\mathbf{o} = [o_1, o_2, \cdots, o_m]^T$$

where the input and output of the network are *d*-dimensional and *m*-dimensional vectors respectively.

If we denote w_{ij} as the weight that connects output neuron *i* with input neuron *j*, the activation value for output neuron i can be written as:

$$v_i = \sum_{j=1}^{d} w_{ij} x_j$$

Note that the activation value is a scalar.

After receiving the activation, neurons perform processing of the activation signal. The processing can be considered as a transform with strong non-linearity:

$$o_i = \varphi(v_i) = \varphi(\sum_{j=1}^{d} w_{ij} x_j)$$

For ease of representation, we define the weight vector as:

$$\mathbf{w}_i = [w_{i1}, w_{i2}, \cdots, w_{id}]$$

Then the activation value to output layer neuron *i* is:

$$v_i = \mathbf{w}_i \mathbf{x}$$

The output of neuron i is given by:

$$o_i = \varphi(v_i) = \varphi(\mathbf{w}_i \mathbf{x})$$

We next introduce a vector operator $\boldsymbol{\varphi}$ that maps the input **x** from input space to output space:

$$\mathbf{o} = \boldsymbol{\varphi}(\mathbf{W}\mathbf{x})$$

where weight matrix **W** is defined as:

$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1d} \\ w_{21} & w_{22} & \cdots & w_{2d} \\ \vdots & \vdots & \vdots & \vdots \\ w_{m1} & w_{m2} & \cdots & w_{md} \end{bmatrix}$$

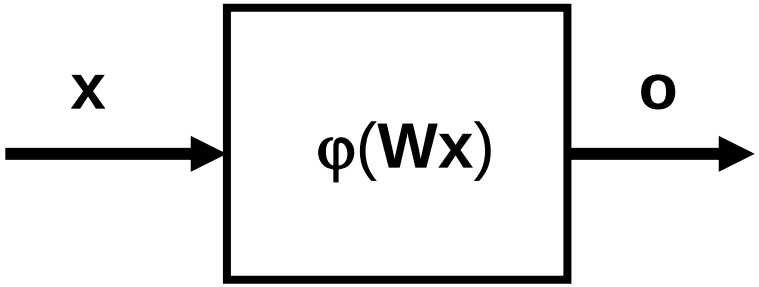
The vector operator $\boldsymbol{\varphi}$ is defined as:

$$\boldsymbol{\varphi}(\bullet) = [\varphi(v_1), \varphi(v_2), \cdots, \varphi(v_m)]^T$$

The mapping from the input to the output is instantaneous, with no time delay. So we have:

$$\mathbf{o} = \varphi(\mathbf{W}\mathbf{x})$$

The block diagram of this realization is as follow:

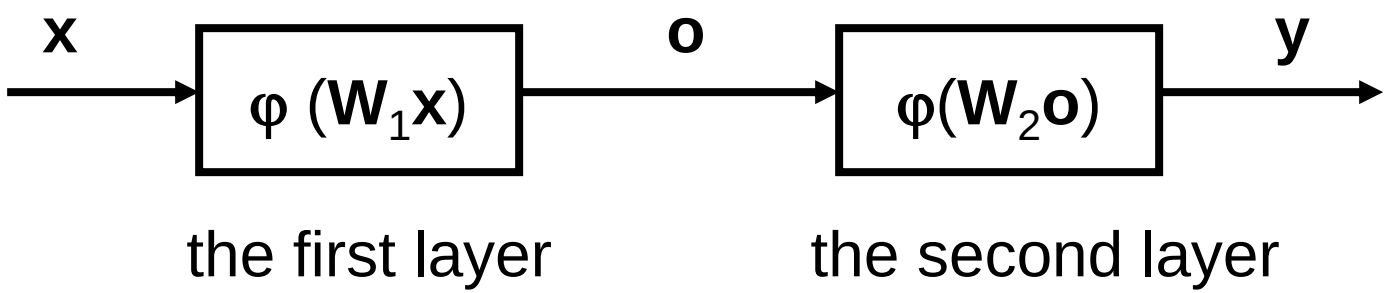


The output vector **o** and input vector **x** are often called output pattern and input pattern respectively.

- **Multi-layer Feed-Forward Neural Network**

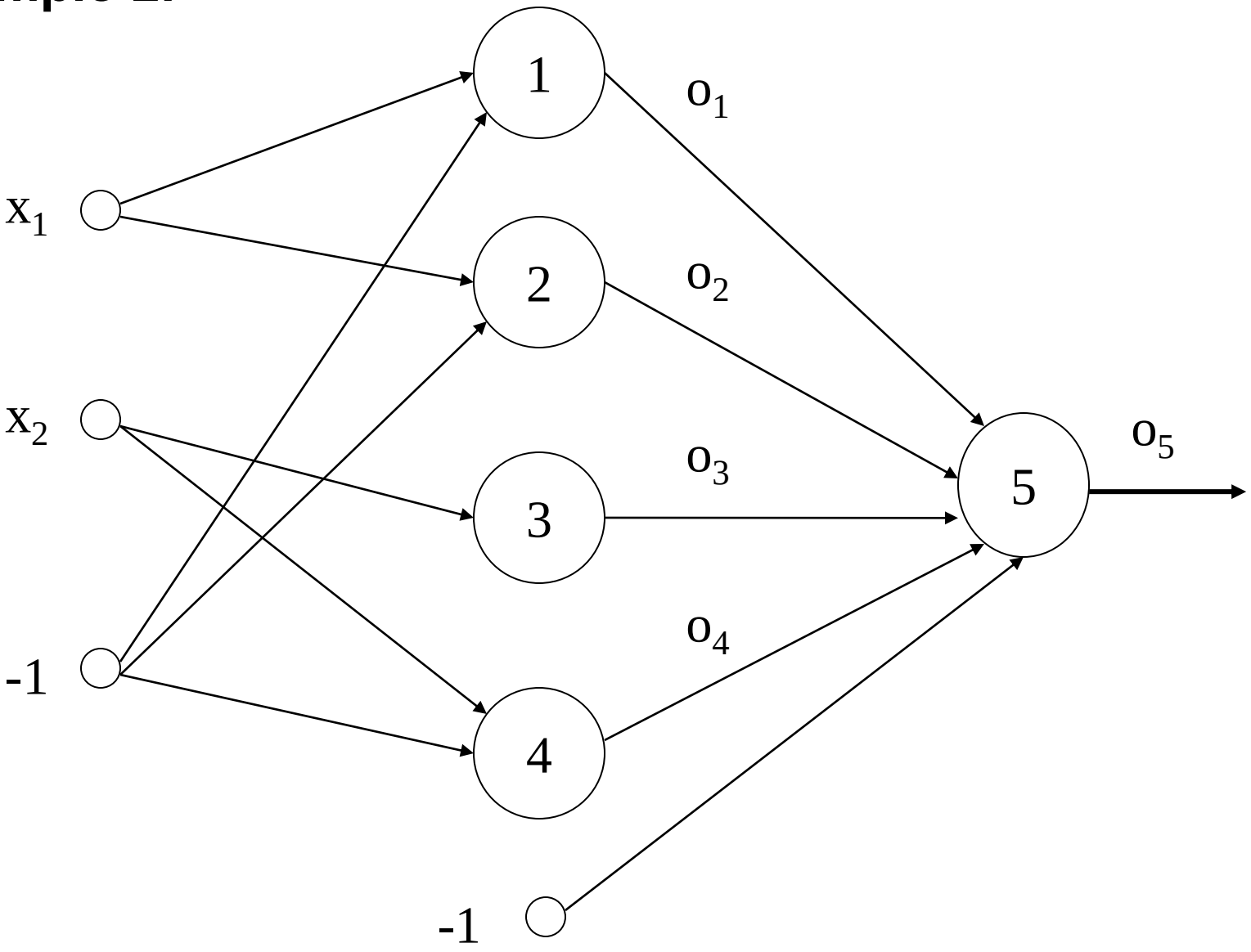
A multi-layer feed-forward neural network consists of two or more connected single-layer feed-forward neural networks. The connection is in series: the output of the preceding layer is used as the input of the following layer.

The diagram of the multi-layer feed forward neural network is shown below:



Where $\mathbf{W}_1$ and $\mathbf{W}_2$ are the weight matrices of the first layer and the second layer respectively.

Example 1:



The connection weight vectors are:

$$\mathbf{w}_1 = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}$$

$$\mathbf{w}_2 = \begin{bmatrix} -1 & 0 & -2 \end{bmatrix}$$

$$\mathbf{w}_3 = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}$$

$$\mathbf{w}_4 = \begin{bmatrix} 0 & -1 & -3 \end{bmatrix}$$

$$\mathbf{w}_5 = \begin{bmatrix} 1 & 1 & 1 & 1 & 3.5 \end{bmatrix}$$

Based on the connections and the weights, we obtain:

$$\begin{aligned} v_1 &= x_1 - 1 \\ v_2 &= -x_1 + 2 \\ v_3 &= x_2 \\ v_4 &= -x_2 + 3 \end{aligned}$$

Then the output of the hidden layer neurons is obtained as:

$$o_1 = \varphi(x_1 - 1)$$
$$o_2 = \varphi(-x_1 + 2)$$
$$o_3 = \varphi(x_2)$$
$$o_4 = \varphi(-x_2 + 3)$$

The output of the hidden layer is then used as the input of the output layer. The output of the overall network is as follow:

$$o_5 = \varphi(o_1 + o_2 + o_3 + o_4 - 3.5)$$

Assuming the activation function is bi-polar binary function:

$$\varphi(\mathrm{v}) = \begin{cases} 1 & \text{if } \mathrm{v} \geq 0 \\ -1 & \text{if } \mathrm{v} < 0 \end{cases}$$

The value of o_5 is 1 if and only if

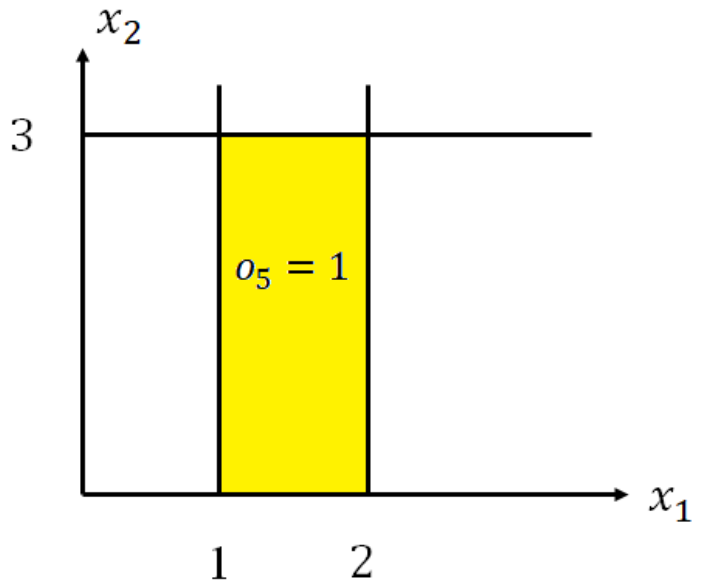
$$o_1 + o_2 + o_3 + o_4 \geq 3.5$$

This actually requires that:

$$o_1 = o_2 = o_3 = o_4 = 1$$

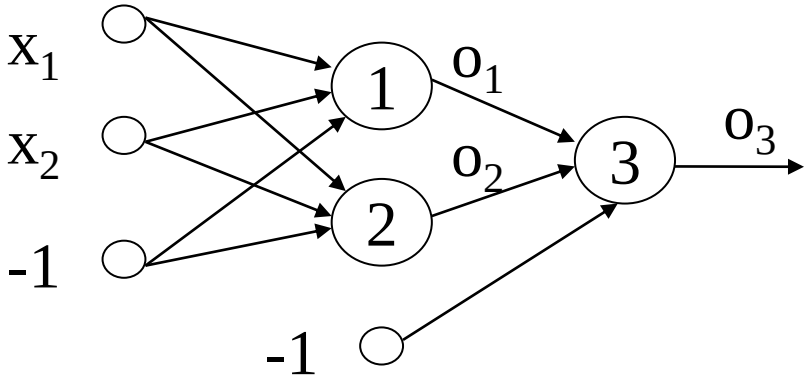
which is equivalent to:

$$x_1 \geq 1$$
$$x_1 \leq 2$$
$$x_2 \geq 0$$
$$x_2 \leq 3$$



Example 2:

Consider the following 2-layer neural network:



The weight vectors are:

$$\mathbf{w}_1 = [1 \quad 1 \quad 1.5] \qquad \mathbf{w}_2 = [1 \quad 1 \quad 0.5] \qquad \mathbf{w}_3 = [-2 \quad 1 \quad 0.5]$$

Assume x_1 and x_2 are in the following range:

$$0 \le x_1 \le 1$$

$$0 \le x_2 \le 1$$

Based on the weight vectors given, we have:

$$v_1 = x_1 + x_2 - 1.5$$

$$v_2 = x_1 + x_2 - 0.5$$

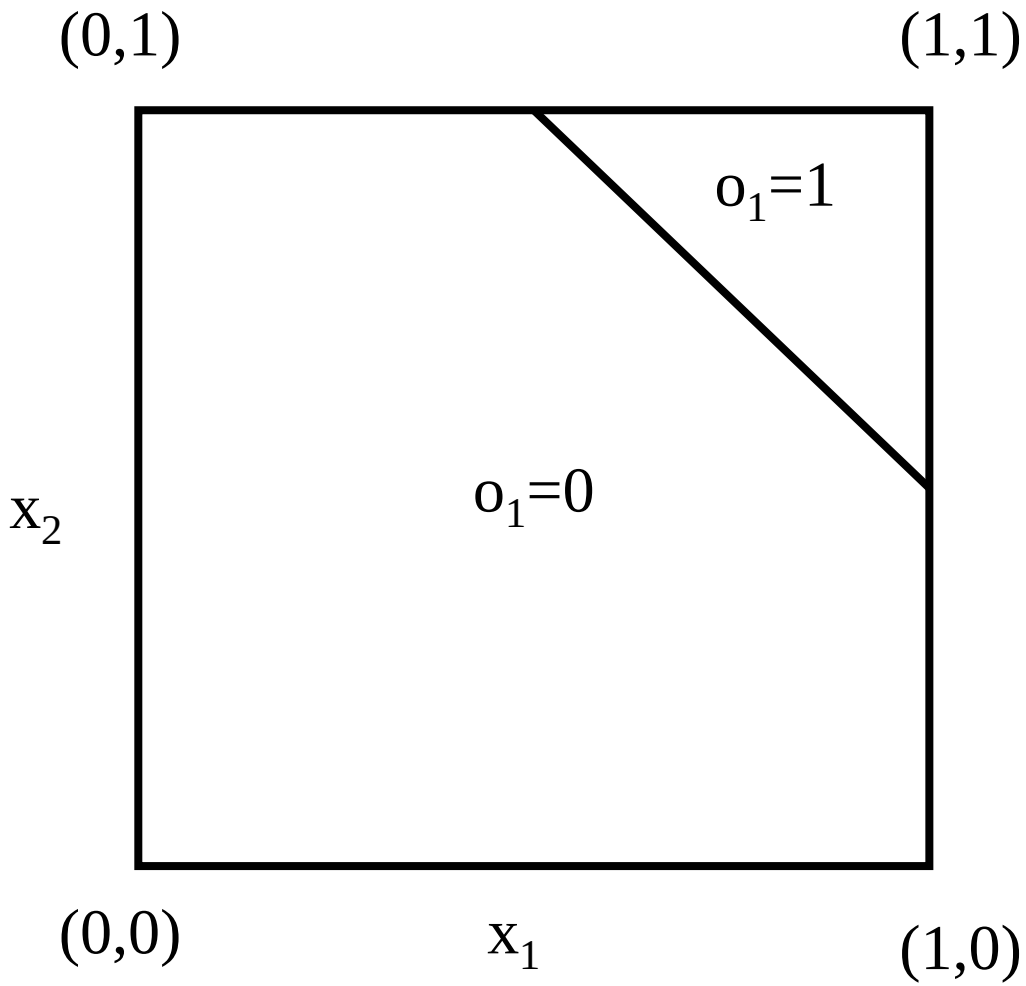
Assume the activation function is uni-polar binary function:

$$\varphi(v) = \begin{cases} 1 & if\ v \geq 0 \\ 0 & if\ v < 0 \end{cases}$$

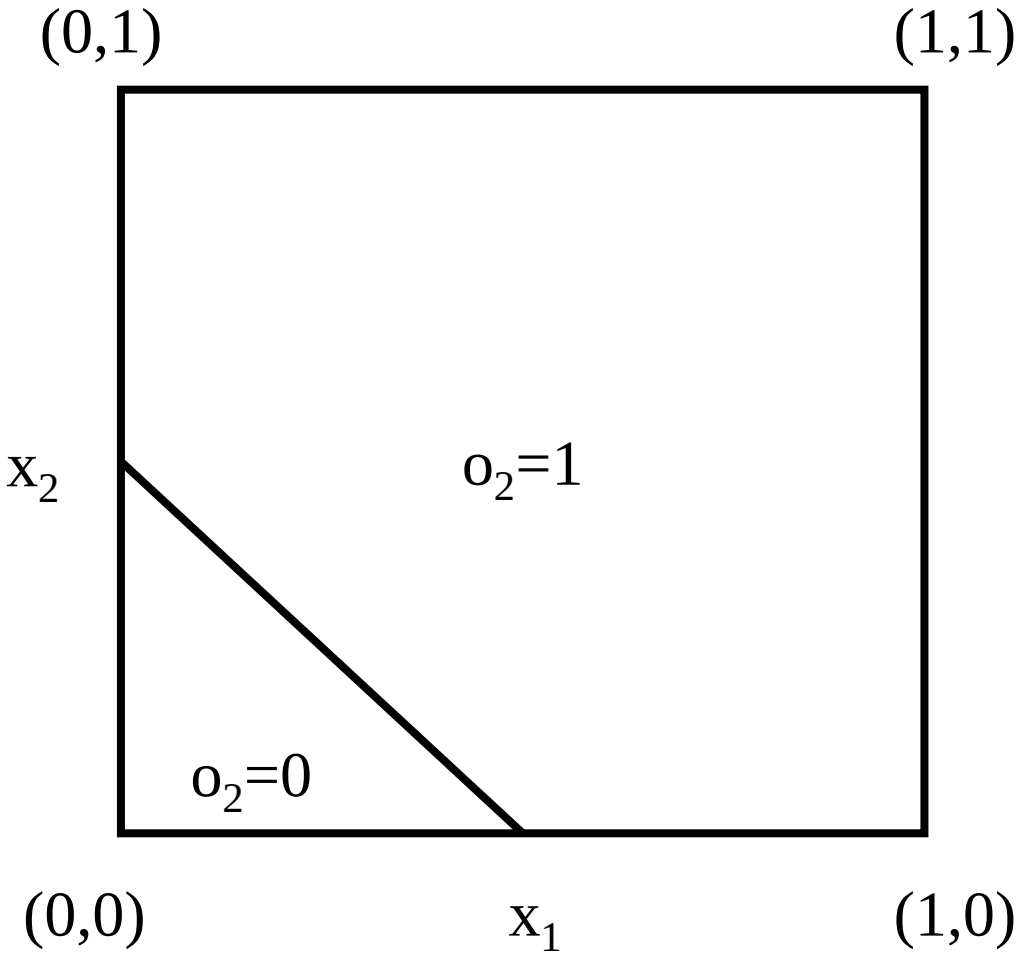
Then we have:

$$o_1 = \begin{cases} 1 & if\ x_1 + x_2 - 1.5 \geq 0 \\ 0 & if\ x_1 + x_2 - 1.5 < 0 \end{cases}$$

$$o_2 = \begin{cases} 1 & if\ x_1 + x_2 - 0.5 \geq 0 \\ 0 & if\ x_1 + x_2 - 0.5 < 0 \end{cases}$$



Output of neuron 1



Output of neuron 2

The activation of neuron 3 is:

$$v_3 = -2o_1 + o_2 - 0.5$$

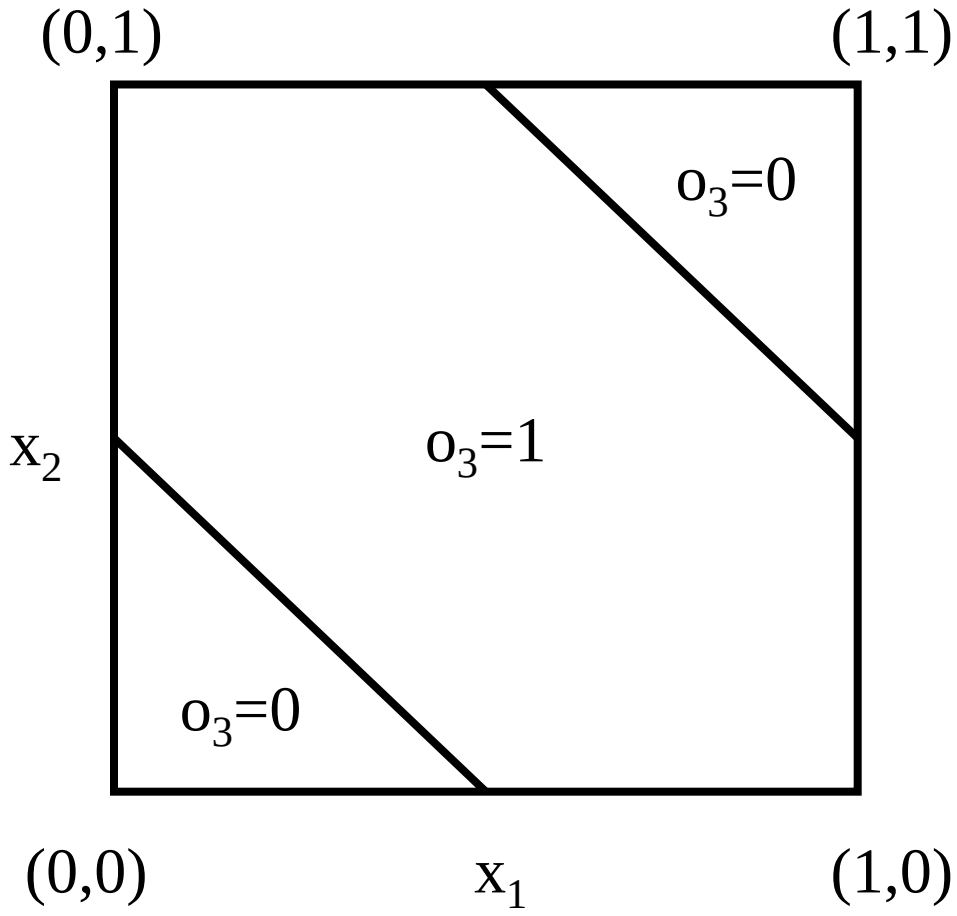
The output of neuron 3 is:

$$o_3 = \begin{cases} 1 & if\ o_2 - 2o_1 - 0.5 \geq 0 \\ 0 & if\ o_2 - 2o_1 - 0.5 < 0 \end{cases}$$

Since the values of o_1 and o_2 are either 1 or 0, thus only when o_1 and o_2 satisfy the following conditions, the value of output o_3 could be 1:

$$\begin{cases} o_1 = 0 \\ o_2 = 1 \end{cases}$$

The region in which o_3=1 is the intersection of the regions where o_1=0 and o_2=1.



Output of the network

If the values of input x_1 and x_2 are either 1 or 0, we can see from the above diagram that:

$$f(1,1) = 0$$
$$f(0,0) = 0$$
$$f(0,1) = 1$$
$$f(1,0) = 1$$

Where *f* is the mapping realized by the above neural network. Actually, f realizes the XOR logic operation:

$$1 \oplus 1 = 0$$
$$0 \oplus 0 = 0$$
$$0 \oplus 1 = 1$$
$$1 \oplus 0 = 1$$

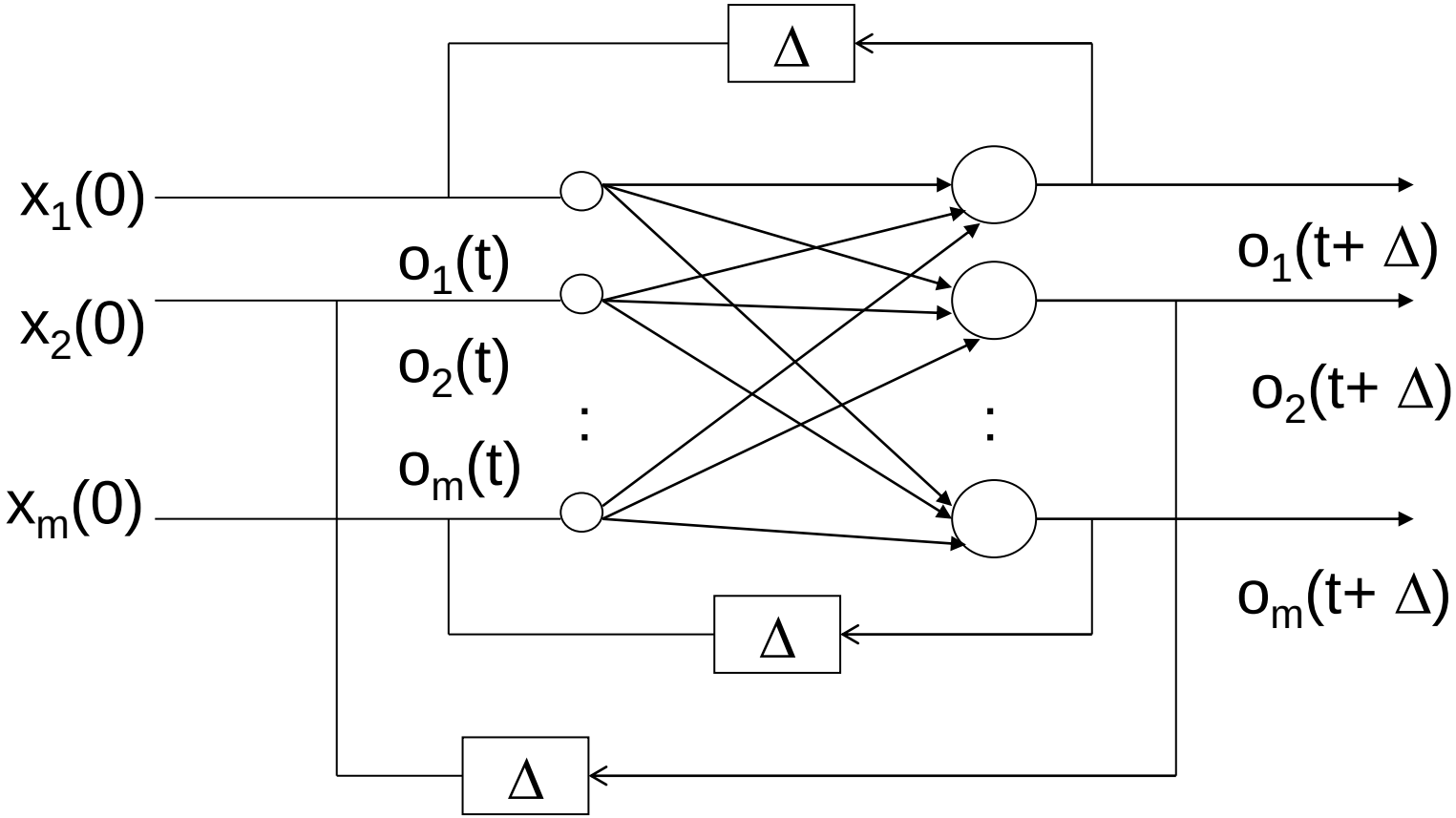
Where $\oplus$ denotes the exclusive OR (XOR) operator.

3. Feedback/Recurrent Neural Networks

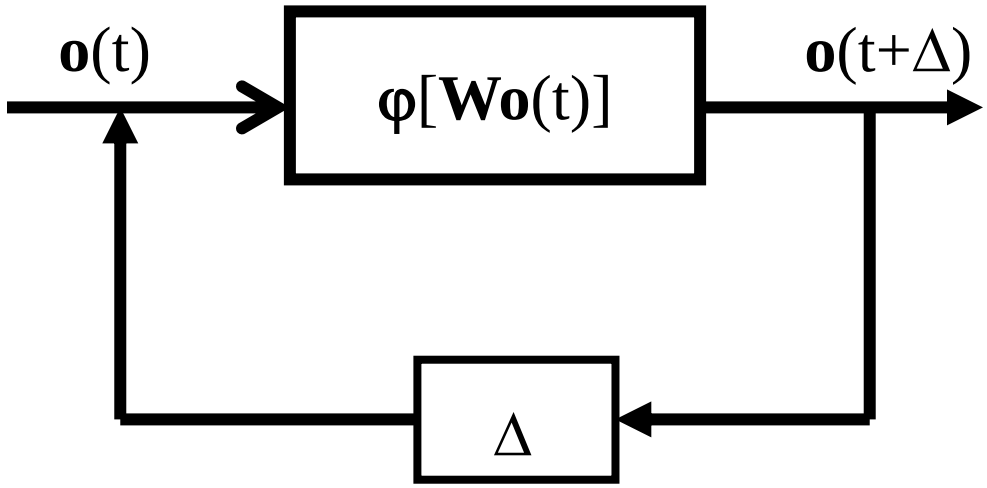
In the multi-layer feed-forward neural network, the layers are connected in the form of cascade, where the outputs of preceding layer neurons are fed to the following layers. The direction of information communication is single and forward: from the preceding layer to the following layer. A feedback neural network is a feed-forward neural network plus a feedback loop which feeds the outputs of output layer neurons back to the input neurons.

The architecture of a typical feedback neural network is shown below:

The architecture of a typical feedback/recurrent neural network is shown below:



The network outputs at the current time instant are influenced by the outputs at previous time. This means the feedback neural network is a dynamic system, and its block diagram is shown below:



The formula of the diagram can be written as:

$$\mathbf{o}(t+\Delta) = \boldsymbol{\varphi}[\mathbf{W}\mathbf{o}(t)]$$

Note that the input is only needed to initialize this network so that **o**(0)=**x**(0). The input is then removed and the system becomes autonomous.

If the time is considered as discrete, and the time delay is an unity delay,

$$\mathbf{t} = \mathbf{n}\Delta$$
$$\Delta = \mathbf{1}$$

where n is an integer. Then the above equation can be rewritten as:

$$\mathbf{o}(\mathrm{n}+1) = \boldsymbol{\varphi}[\mathbf{W}\mathbf{o}(\mathrm{n})]$$

The above discrete-time model of the single layer feedback network can be expanded to a series of nested solutions:

$$\mathbf{o}(1) = \boldsymbol{\varphi}[\mathbf{W}\mathbf{x}(0)]$$

$$\mathbf{o}(2) = \boldsymbol{\varphi}[\mathbf{W}\mathbf{o}(1)] = \boldsymbol{\varphi}[\mathbf{W}[\boldsymbol{\varphi}[\mathbf{W}\mathbf{x}(0)]]]$$

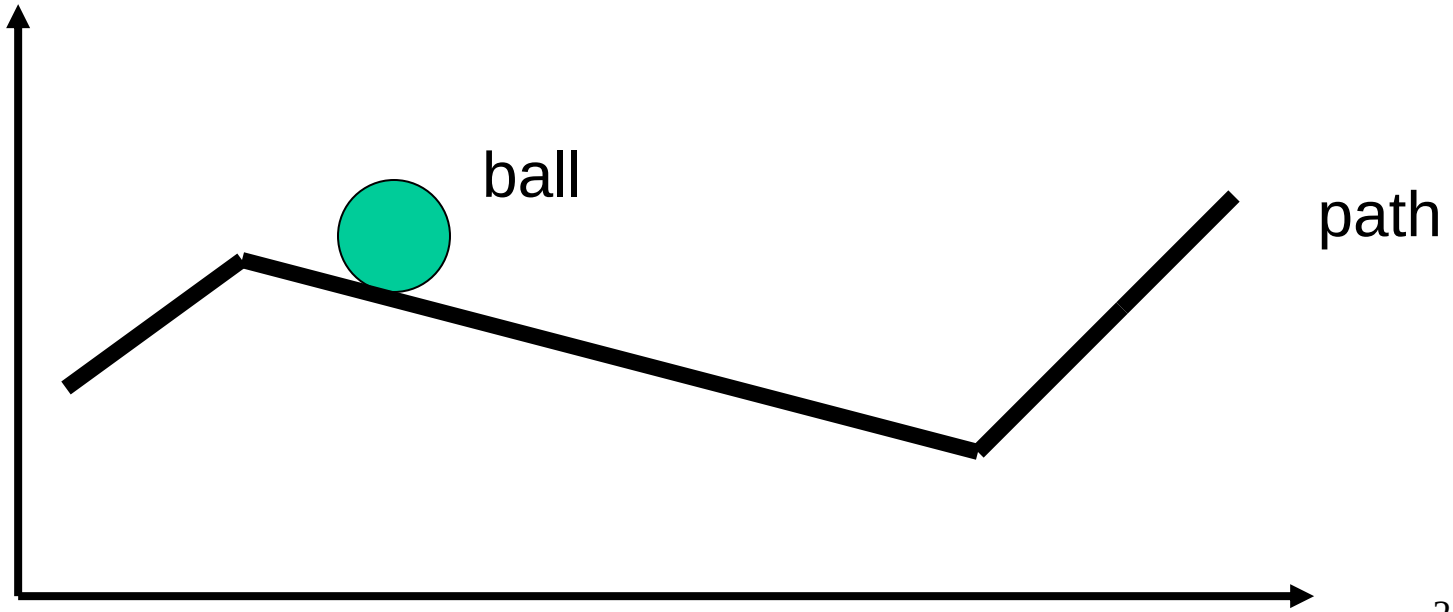
$$\vdots$$

$$\mathbf{o}(\mathrm{n}) = \boldsymbol{\varphi}[\mathbf{W}\boldsymbol{\varphi}[\cdots\boldsymbol{\varphi}[\mathbf{W}\mathbf{x}(0)]\cdots]]$$

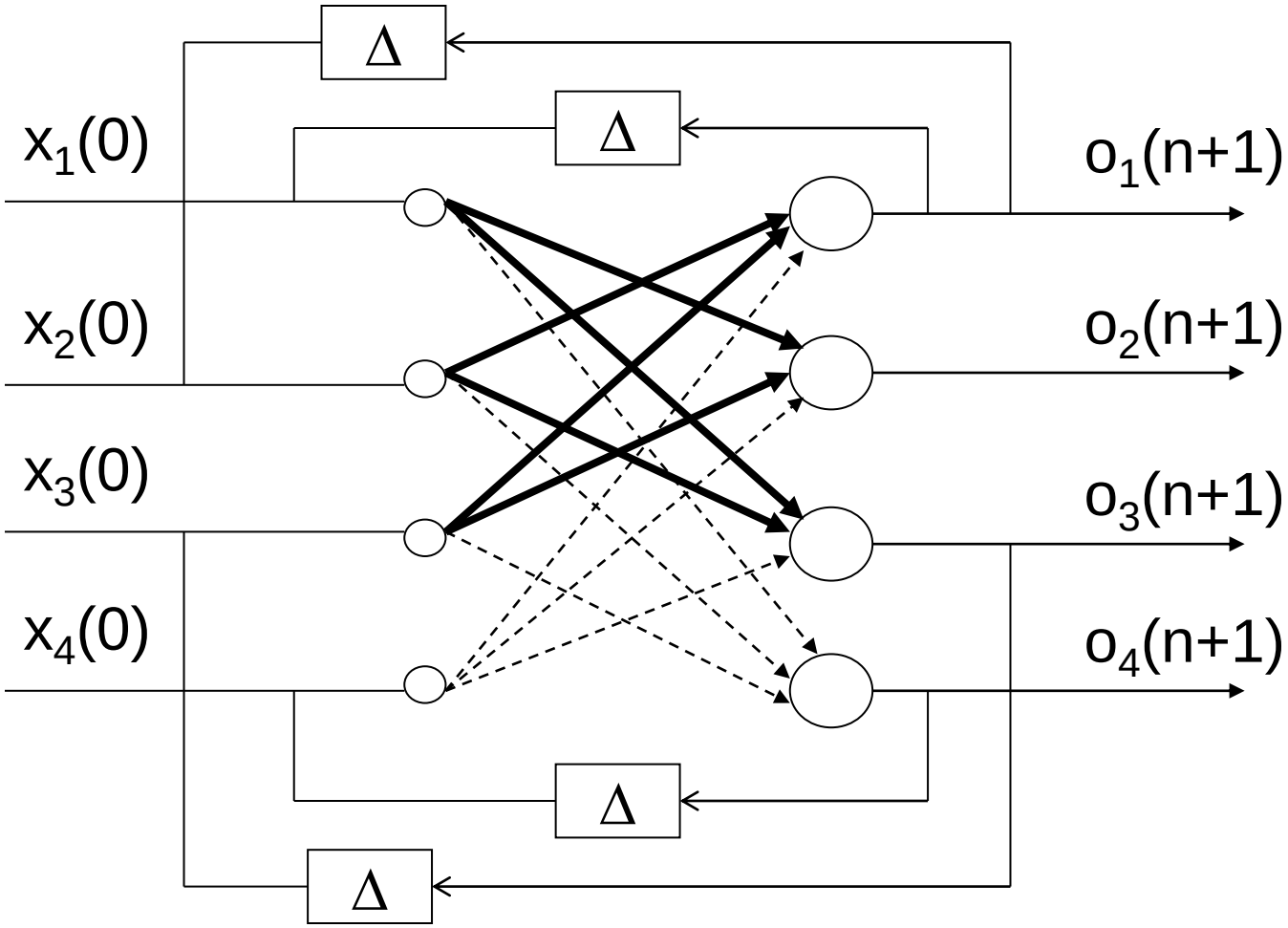
Obviously, the output at time instant *n* depends on the entire history of the network starting from *n*=0. The above network is therefore called recurrent neural network.

The above equations describe the outputs of the network at time instants n=1,2,3,…N. The output vector can be considered as the state variables of the network, these equations therefore describe state transitions.

The network begins state transition once the initial state is given. The transition goes on until it reaches to a stable state, called the equilibrium state, as illustrated below:



Example:



The solid lines denote weight to be 1, the dashed lines to be -1. By inspecting the network, we obtain the following matrix of weight:

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

If we begin with the following initial setting:

$$\mathbf{x}(0) = \begin{bmatrix} 1 & 1 & 1 & -1 \end{bmatrix}^T$$

The output vector at time instant n=1 is:

$$\mathbf{o}(1) = \varphi[\mathbf{W}\mathbf{x}(0)] = \varphi(\begin{bmatrix} 3 & 3 & 3 & -3 \end{bmatrix}^T) = \begin{bmatrix} 1 & 1 & 1 & -1 \end{bmatrix}^T$$

To compute the state at time instant *n*=2, we can take the output at time instant *n*=1 as the initial state setting:

$$\mathbf{o}(2) = \varphi[\mathbf{W}\mathbf{o}(1)] = \begin{bmatrix} 1 & 1 & 1 & -1 \end{bmatrix}^T = \mathbf{o}(1)$$

This means that no state transition occurs, and $[1\ 1\ 1\ -1]^T$ is an equilibrium state. If we start with another initial state:

$$\mathbf{x}(0) = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}^T$$

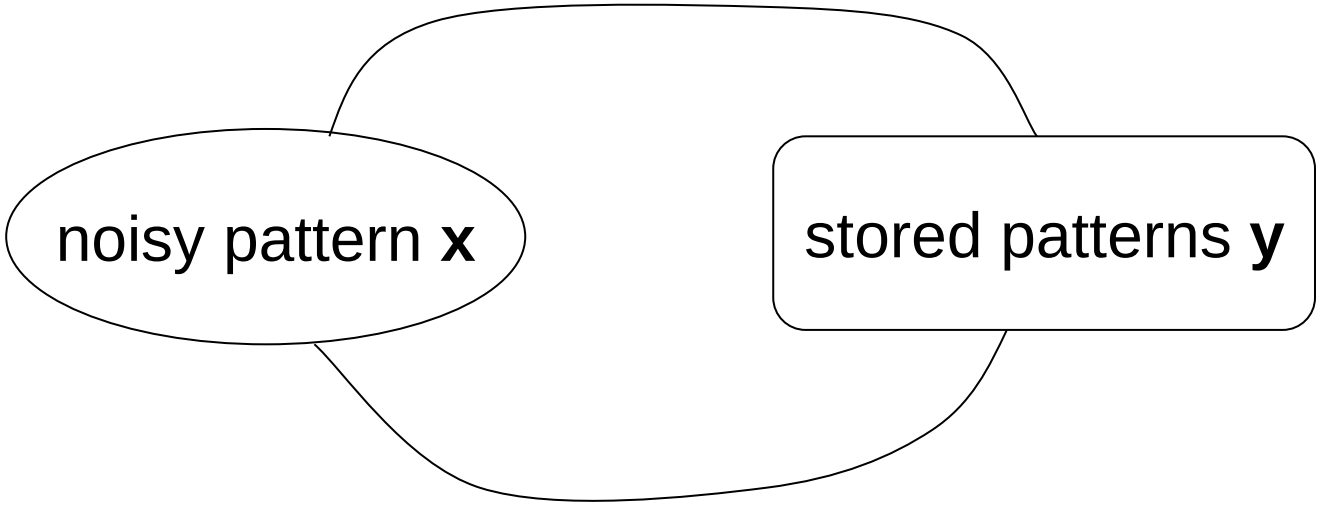
The output at time instant n=1 is:

$$\mathbf{o}(1) = \varphi(\begin{bmatrix} 1 & 1 & 1 & -3 \end{bmatrix}^T) = \begin{bmatrix} 1 & 1 & 1 & -1 \end{bmatrix}^T$$

The output transits from a non-equilibrium to an equilibrium state. Once the state reaches an equilibrium state, no transitions would occur unless external input is applied.

- **Hopfield Network**

Hopfield network is a typical example of the recurrent network that embodies a profound physical principle of storing information in a dynamically stable configuration. The Hopfield network can also be considered as a nonlinear associative memory, the function of which is to retrieve a pattern stored in memory in response to the presentation of an incomplete or noisy version of that pattern.



- **Characteristics of the Hopfield Network**

(1) Single layer;

(2) The output of each neuron is fed back to the input of other neurons except itself;

(3) The network uses bi-polar binary function as its activation function.

The Hopfield network uses bi-polar binary activation function, thus each such neuron has two states described by the level of activation. The "on" state of a neuron is denoted by the output +1, while the "off" state of a neuron is represented by –1. For a network made up of 4 neurons, a typical example of the state of the network is:

$$\begin{bmatrix} 1 & 1 & -1 & 1 \end{bmatrix}^T$$

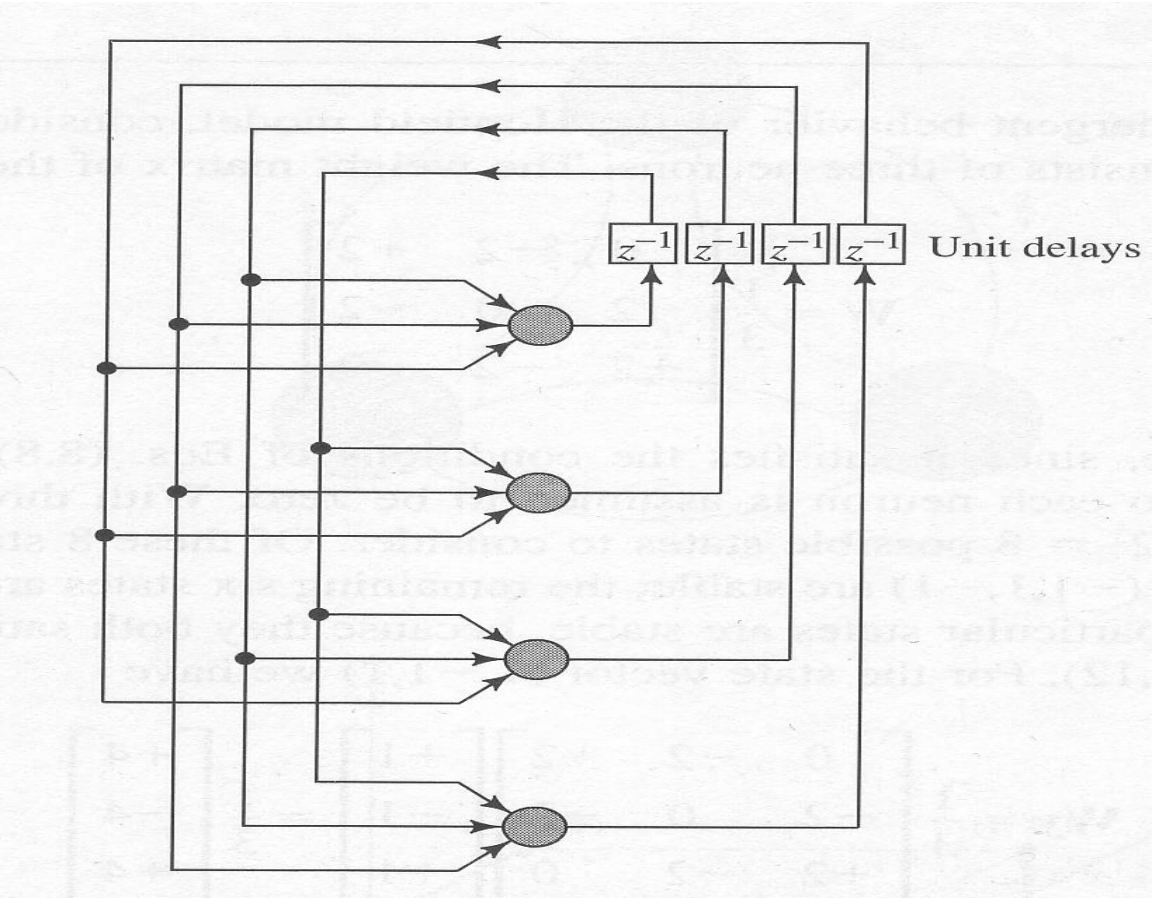


Diagram of the Hopfield network (4 neurons)

Example:

Assume the weight matrix is as follow:

$$\mathbf{W} = \frac{1}{3}\begin{bmatrix} 0 & -2 & 2 \\ -2 & 0 & -2 \\ +2 & -2 & 0 \end{bmatrix}$$

And the initial state is:

$$\mathbf{x}(0) = \begin{bmatrix} 1 & -1 & 1 \end{bmatrix}^T$$

The output at time instant n=1 is then given by:

$$\begin{aligned}\mathbf{o}(1) &= \varphi[\mathbf{W}\mathbf{x}(0)] \\ &= \varphi(\frac{1}{3}\begin{bmatrix}4 & -4 & 4\end{bmatrix}^T) \\ &= \begin{bmatrix}1 & -1 & 1\end{bmatrix}^T \\ &= \mathbf{x}(0)\end{aligned}$$

This means that the state [1,-1,1] is a stable state.

Let's consider another initial state:

$$\mathbf{x}(0) = \begin{bmatrix} -1 & 1 & -1 \end{bmatrix}^T$$

The output at time instant *n*=1:

$$\mathbf{o}(1) = \varphi[\mathbf{W}\mathbf{x}(0)] = \varphi(\frac{1}{3}\begin{bmatrix} -4 & 4 & -4 \end{bmatrix}^T) = \begin{bmatrix} -1 & 1 & -1 \end{bmatrix}^T$$

The output at time instant *n*=2:

$$\mathbf{o}(2) = \varphi[\mathbf{W}\mathbf{o}(1)] = \begin{bmatrix} -1 & 1 & -1 \end{bmatrix}^T = \mathbf{o}(1)$$

Obviously, state transition would no loner occur once the state of the network reaches $[-1\ 1\ -1]^T$. Hence $[-1\ 1\ -1]^T$ is a stable state.

The network has 3 outputs, and each output has 2 states, the total number of states of the network is 8:

(+1 -1 -1), (+1 -1 1), (+1 +1 -1), (+1 +1 +1),

(-1 -1 -1), (-1 -1 +1), (-1 +1 -1), (-1 +1 +1)

Of the 8 states, only two states (1 –1 1) and (-1 1 –1) are stable, the remaining 6 states are all unstable.

For a general case, the condition of stability of a state is:

$$\mathbf{o}(n+1) = \phi[\mathbf{W}\mathbf{o}(n)] = \mathbf{o}(n)$$

For a stable network, any state will eventually move to a stable state.

- **Energy function**

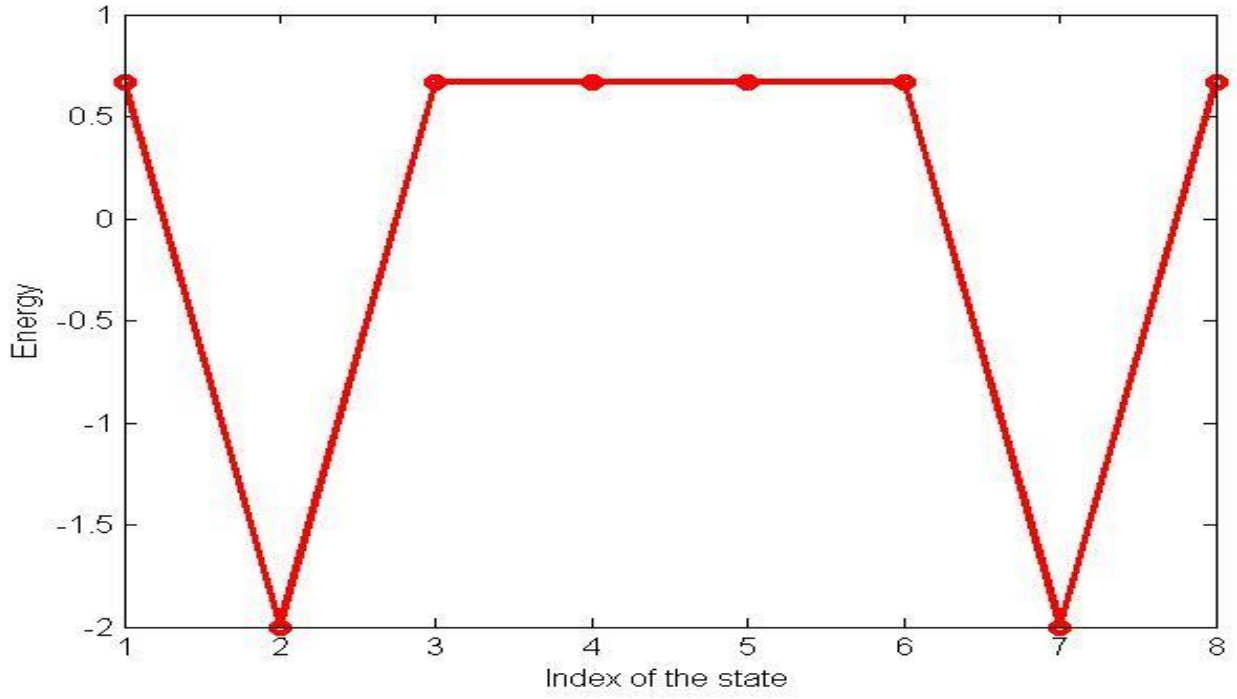
Considerable insight into the Hopfield neural network can be gained by evaluating its respective energy function, which is defined as:

$$E(\mathbf{o}) = -\frac{1}{2}\mathbf{o}^T\mathbf{W}\mathbf{o}$$

Based on the above energy definition, we evaluate the energy of the 8 states. For example, for state [1 -1 -1], the corresponding energy is:

$$E = -\frac{1}{2}\times\begin{bmatrix}1 & -1 & -1\end{bmatrix}\times\frac{1}{3}\begin{bmatrix}0 & -2 & 2\\ -2 & 0 & -2\\ +2 & -2 & 0\end{bmatrix}\times\begin{bmatrix}1\\ -1\\ -1\end{bmatrix} = 2/3$$

Similarly, we can evaluate the energy of the remaining 7 states. The energies of the eight states are shown below:



- **Weight determination—Training nonlinear mapping using linear solution technique**

The presence of a nonlinear activation function does not necessarily require a nonlinear solution formation. In Hebbian learning, the correlation between the pre-synaptic and post-synaptic signals have been used for weight determination. Next, we will develop a procedure to form weight matrix **W** based on the above idea.

Consider a single stimulus-response pair (**s**, **r**), with the d-dimensional stimulus vector **s** normalized to unity length:

$$\|\mathbf{s}\| = \sqrt{\mathbf{s}^T\mathbf{s}} = 1$$

The formation of a weight matrix using the following form is proposed:

$$\mathbf{W} = \mathbf{r}\mathbf{s}^T$$

Next, we examine the storage property of such a weight formation. For an arbitrary stimulus vector **s**, the activation signals of the neurons are given by:

$$\mathbf{W}\mathbf{s} = \mathbf{r}\mathbf{s}^T\mathbf{s} = \mathbf{r}$$

Thus, the output of the Hopfield neural network is obtained by applying the bi-polar binary function on the activation signals:

$$\mathbf{o} = \varphi(\mathbf{W}\mathbf{s}) = \varphi(\mathbf{r}) = \mathbf{r}$$

Next, we examine the storage property under a distorted stimulus. Assuming the distorted version of the stimulus is denoted by **s',** the activation signals of the neurons are given by:

$$\mathbf{W}\mathbf{s}' = \mathbf{r}\mathbf{s}^T\mathbf{s}' = (\mathbf{s}^T\mathbf{s}')\mathbf{r}$$

For normalized **s** and **s'**, we can verify that:

$$\mathbf{s}^T\mathbf{s}' < \mathbf{s}^T\mathbf{s} = 1$$

(i) Case 1:

$$0 \leq \mathbf{s}^T\mathbf{s}' < 1$$

Then the output is:

$$\mathbf{o} = \varphi(\mathbf{W}\mathbf{s}) = \varphi[(\mathbf{s}^T\mathbf{s}')\mathbf{r}] = \mathbf{r}$$

(ii) Case 2:

$$\mathbf{s}^T\mathbf{s}' < 0$$

Then the output is:

$$\mathbf{o} = \varphi(\mathbf{W}\mathbf{s}) = \varphi[(\mathbf{s}^T\mathbf{s}')\mathbf{r}] \neq \mathbf{r}$$

This indicates the generalization capability of Hopfield neural network under this learning algorithm. If the stimulus is seriously distorted, the Hopfield neural network may not restore it.

Suppose N d-dimensional vectors, also called fundamental memories, are to be stored. The N fundamental memories are denoted by:

$$\{\mathbf{s}_i \mid i = 1,2,\cdots,N\}$$

In the normal operation of the Hopfield network, the stimulus and the response are the same, and the neurons have no self-feedback. We set:

$$\mathbf{r}_i = \mathbf{s}_i$$

$$w_{ii} = 0$$

The matrix equation is given by:

$$\mathbf{W} = \frac{1}{d}\sum_{i=1}^{N}\mathbf{s}_i\mathbf{s}_i^T - \frac{N}{d}\mathbf{I}$$

where **I** denotes a $d \times d$ identity matrix

- **Operational Features of the Hopfield Network**

There are two phases in the operation of the Hopfield network: the storage phase and the retrieval phase.

❑ Storage Phase (Design Phase)

- The number of neurons is equal to the dimension of vectors to store.
- The output of each neuron is fed back to all other neurons. There is no self-feedback in the network.
- The weight matrix of the network is symmetric, and is given by:

$$\mathbf{W} = \frac{1}{d}\sum_{i=1}^{N}\mathbf{s}_i\mathbf{s}_i^T - \frac{N}{d}\mathbf{I}$$

❑ Retrieval Phase

During the retrieval phase, an d-dimensional vector **x**, called probe, is imposed on the Hopfield network as its input. The vector has elements +1 or –1. Often the probe is an incomplete or noisy version of a fundamental memory. Information retrieval then proceeds in accordance with dynamic rule, in which the state of the network is updated at each iteration. The updating of the state is continued until no further changes occur:

$$o_j(n+1) = \varphi[\sum_{i=1}^{n} w_{ji} o_i(n)]$$

If we write the above equation in matrix form, we have:

$$\mathbf{o}(n+1) = \varphi[\mathbf{W}\mathbf{o}(n)]$$

❑ Summary of the Hopfield network

The operation procedure for the Hopfield network may now be summarized as follows:

Step 1: Construct a Hopfield structure that has *d* neurons, where *d* is the number of the dimension of vectors

Step 2: Learn the weights. Assume the *N* vectors are:

$$\{\mathbf{s}_i \mid i = 1,2,\cdots,N\}$$

$$\mathbf{s}_i = [s_{1i}, s_{2i}, \cdots, s_{di}]^T$$

Then the weight matrix is given by:

$$\mathbf{W} = \frac{1}{d}\sum_{i=1}^{N}\mathbf{s}_i\mathbf{s}_i^T - \frac{N}{d}\mathbf{I}$$

Step 3: Initialization. The initial state of the Hopfield network can be set to the input pattern:

$$\mathbf{o}(0) = \mathbf{x}$$

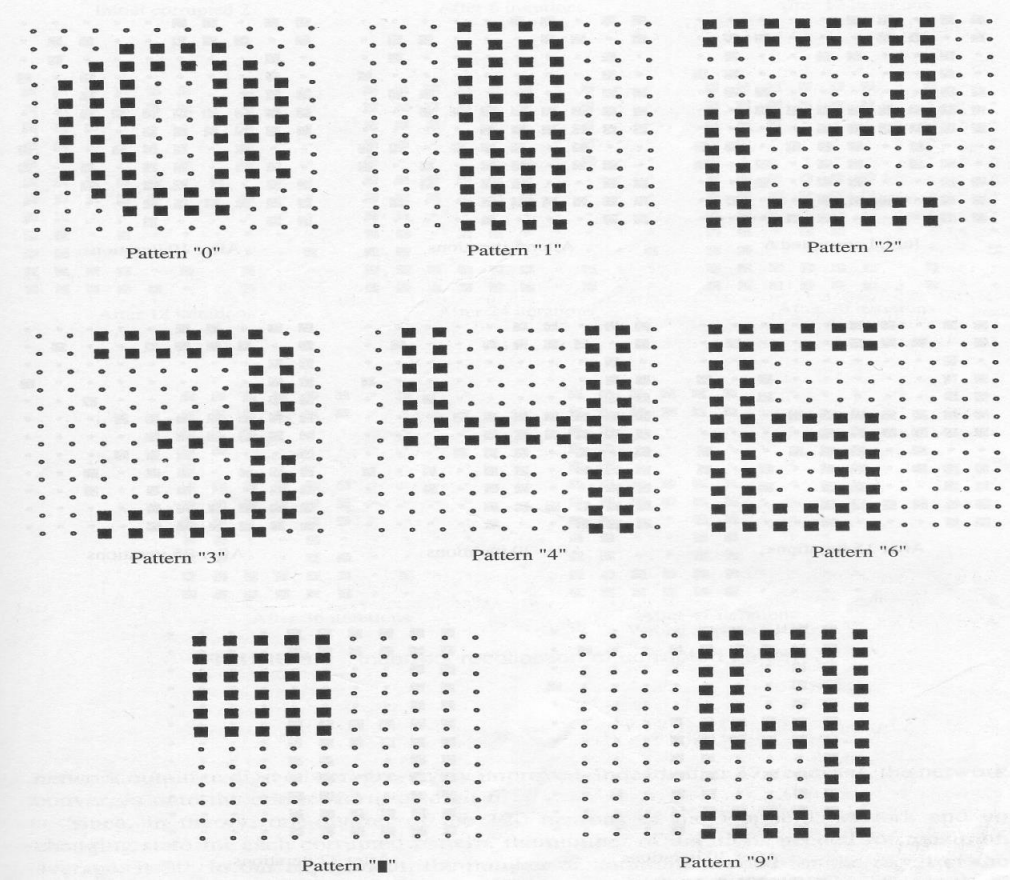
Step 4: Iteration. Update the output of the network according to the following rule:

$$\mathbf{o}(n+1) = \varphi[\mathbf{W}\mathbf{o}(n)]$$

Step 5: Output. The iteration at Step 4 is continued until the state of the network is unchanged. Assume totally L iterations are repeated, the resulting output of the network is given by:

$$\mathbf{y} = \mathbf{o}(\mathrm{L})$$

Example



Totally 8 patterns 0,1,2,3,4,6,..,9

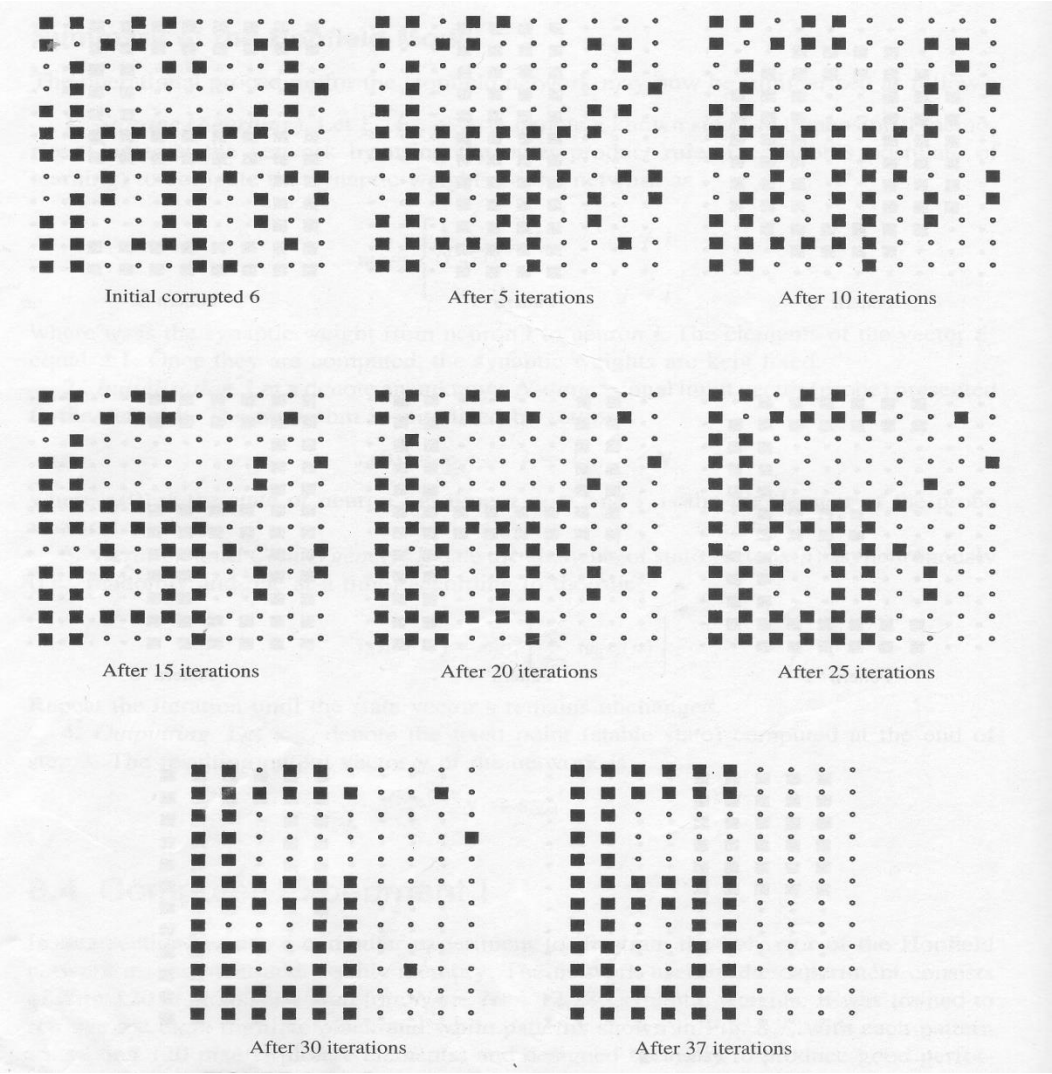
Each number is represented by 120 points, therefore the pattern vector is 120-dimensional, and network should have 120 neurons.

Procedure to design and retrieve:

Step 1: Define a Hopfield network with 120 neurons

Step 2: Compute the weight matrix of the network using the 8 patterns (fundamental memories) provided

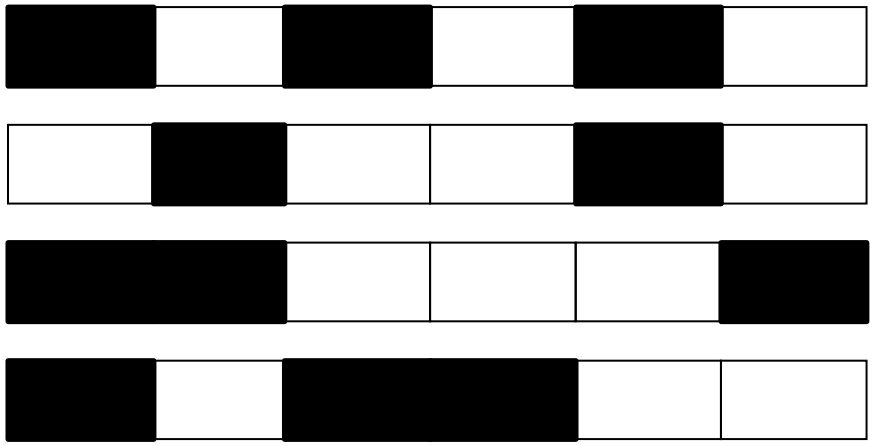
Step 3: Retrieve. Let the initial state be the pattern to be retrieved. Repeatedly update the state of the network until the state is stabilized.



Iteration process

- **Storage Capacity of the Hopfield Network**

Let's first investigate the following example. Assume we have four 6-pixel binary images as shown below:



(a)

(b)

(c)

(d)

Assume the black pixel and the white pixel are represented by +1 and -1 respectively. Design a Hopfield network to store the 4 images.

The four images can be represented by the following 4 vectors:

$$\mathbf{s}_1 = \begin{bmatrix} 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix}^T$$

$$\mathbf{s}_2 = \begin{bmatrix} -1 & 1 & -1 & -1 & 1 & -1 \end{bmatrix}^T$$

$$\mathbf{s}_3 = \begin{bmatrix} 1 & 1 & -1 & -1 & -1 & 1 \end{bmatrix}^T$$

$$\mathbf{s}_4 = \begin{bmatrix} 1 & -1 & 1 & 1 & -1 & -1 \end{bmatrix}^T$$

Weight matrix is obtained as:

$$\mathbf{W} = \frac{1}{3} \begin{bmatrix} 0 & -1 & 1 & 0 & -1 & 0 \\ -1 & 0 & -2 & -1 & 0 & 1 \\ 1 & -2 & 0 & 1 & 0 & -1 \\ 0 & -1 & 1 & 0 & -1 & 0 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & 1 & -1 & 0 & -1 & 0 \end{bmatrix}$$

Consider an input pattern $\mathbf{x}=\mathbf{s}_1$, we have:

$$\mathbf{o}(1) = \varphi(\mathbf{W}\mathbf{x}) = \begin{bmatrix} 1 & -1 & 1 & 1 & 1 & -1 \end{bmatrix}^T$$

$$\mathbf{o}(2) = \varphi[\mathbf{W}\mathbf{o}(1)] = \begin{bmatrix} 1 & -1 & 1 & 1 & -1 & -1 \end{bmatrix}^T = \mathbf{s}_4$$

$$\mathbf{o}(3) = \varphi[\mathbf{W}\mathbf{o}(2)] = \begin{bmatrix} 1 & -1 & 1 & 1 & -1 & -1 \end{bmatrix}^T = \mathbf{s}_4$$

Obviously, $\mathbf{x}$ converges to $\mathbf{s}_4$ rather than $\mathbf{s}_1$.

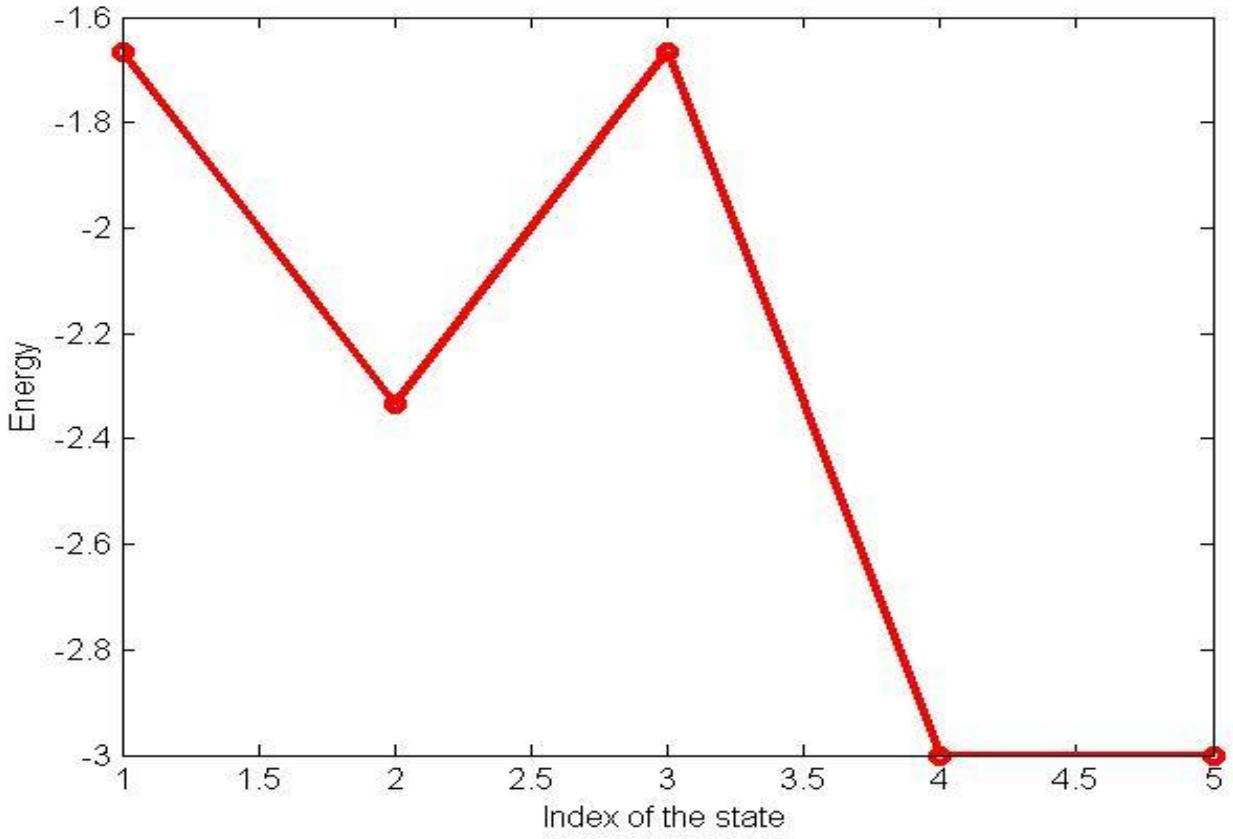
Consider another input pattern $\mathbf{x}=\mathbf{s}_2$, we have:

$$\mathbf{o}(1) = \varphi(\mathbf{W}\mathbf{x}) = \begin{bmatrix} -1 & 1 & -1 & -1 & 1 & 1 \end{bmatrix}^T$$

$$\mathbf{o}(2) = \phi[\mathbf{W}\mathbf{o}(1)] = \begin{bmatrix} -1 & 1 & -1 & -1 & 1 & 1 \end{bmatrix}^T$$

Obviously, $\mathbf{x}$ will converge to a state that is not a fundamental memory of the Hopfield network built, i.e. a spurious state

The energies corresponding to the 4 states and the spurious state are shown below:



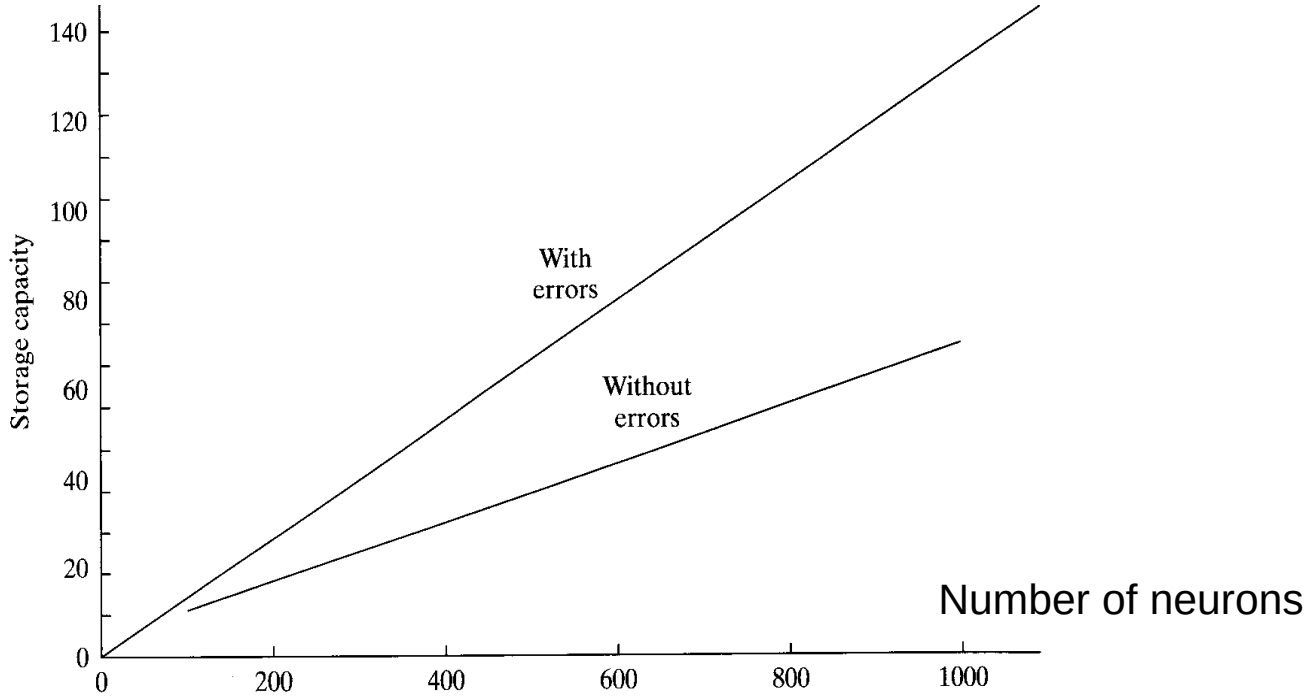
The above example reveals two points relating to the storage capacity problem:

(1) The fundamental memories of a Hopfield network are not always stable;
(2) Spurious state representing other stable states that are different from the fundamental memories can arise.

These two phenomena tend to decrease the efficiency of storage of the Hopfield neural network. Studies show that the quality of memory retrieval deteriorates with the increasing load parameter, which is defined as follow:

$$\alpha = N / d$$

Where *N* is the number of fundamental memories and *d* is the number of neurons of the Hopfield network



Conclusions:

(1) Storage capacity of the Hopfield network scales linearly with the size of network;

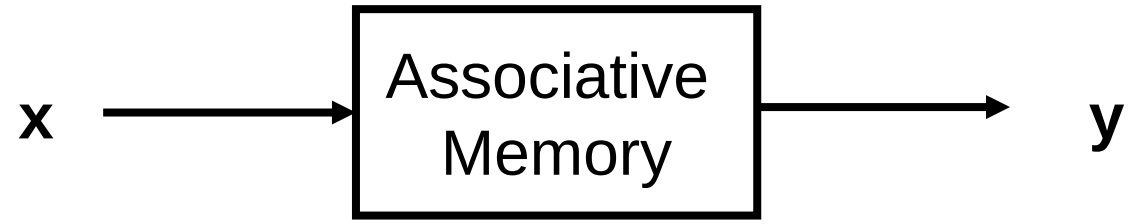
(2) A major limitation of the Hopfield network is that its storage capacity must be maintained small for the fundamental memories to be recoverable.

4. Bi-directional Associative Memories

Recurrent neural network can be used to store information. The is because the dynamic behavior of recurrent neural network exhibits stable state. The iteration procedure actually moves the state from the initial point to the fundamental memory. The fundamental memory is also called stored memory.

Neural networks of this class are called associative memories. The recurrent neural network is an example of associative memories. An efficient associative memory can store a large set of patterns as memories.

Essentially, the associative memory is a kind of mapping:



which maps the input from a *d*-dimensional space to a m-dimensional space:

$$\mathbf{x} \in R^d \rightarrow \mathbf{y} \in R^m$$

The Hopfield neural network just has one layer. Next we introduce bi-directional associative memories (BAM), which consists of two layers and transmits information in both forward and backward directions.

- **Bi-directional Associative Memory (BAM)**

  Bi-directional associative memory consists of two layers. It uses the forward and backward information to produce an associative search for stored association.

  Assume *N* association pairs are stored in the memory:

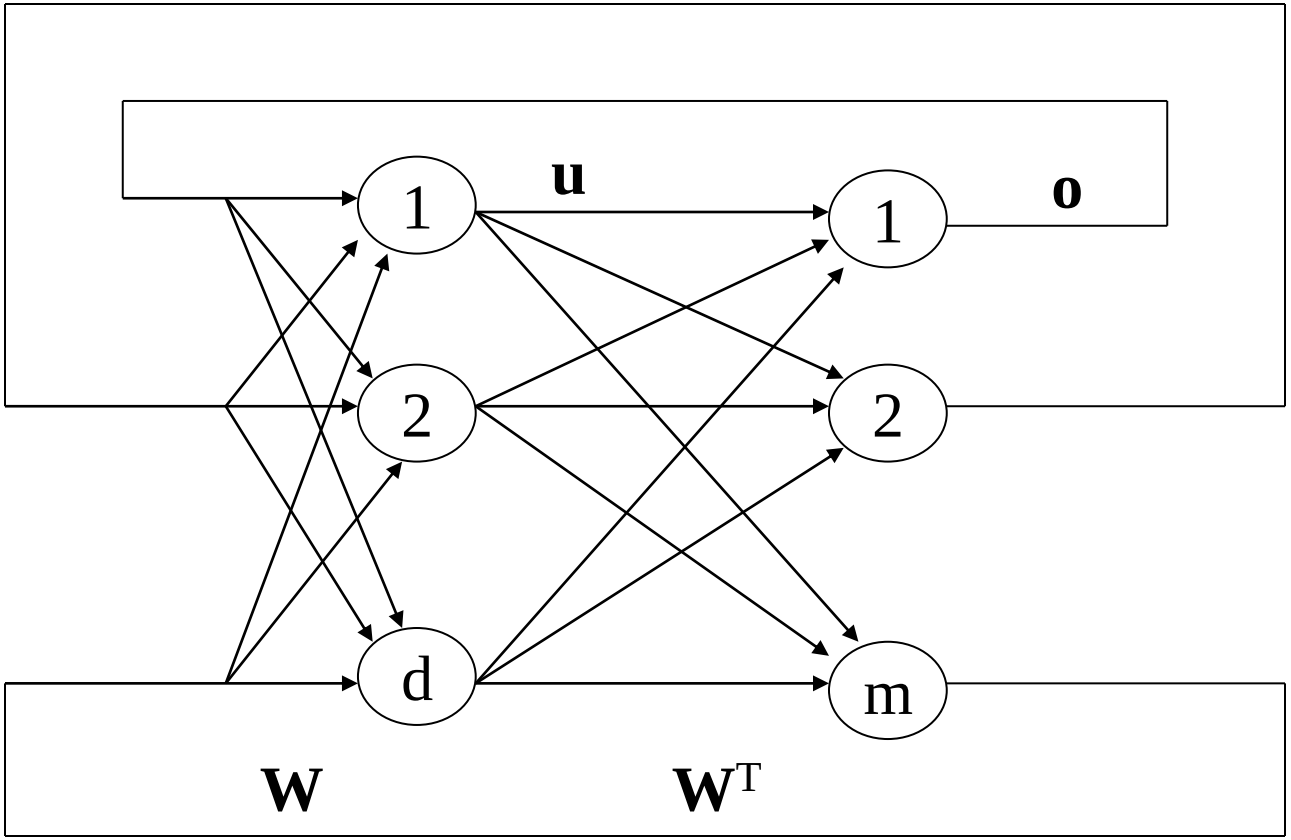
$$(\mathbf{x}_1, \mathbf{y}_1) \quad (\mathbf{x}_2, \mathbf{y}_2) \quad \cdots \quad (\mathbf{x}_N, \mathbf{y}_N)$$

  Where **x** and **y** are *d*-dimensional and *m* dimensional vectors:

$$\mathbf{x}_k = [x_{k1}, x_{k2}, \cdots, x_{kd}]^T$$

$$\mathbf{y}_k = [y_{k1}, y_{k2}, \cdots, y_{km}]^T$$

The architecture of a BAM



Layer 1 Layer 2

Assume the activation functions of the two layer neurons are bipolar binary. The output of neuron i of the first layer is:

$$u_i = \varphi(\sum_{j=1}^{m} w_{ij} o_j)$$

The output of neuron j of the second layer is:

$$o_j = \varphi(\sum_{i=1}^{n} w_{ij} u_i)$$

In vector form:

$$\mathbf{u} = \varphi(\mathbf{W}\mathbf{o})$$

$$\mathbf{o} = \varphi(\mathbf{W}^T\mathbf{u})$$

Where:

$$\mathbf{u} = [u_1, u_2, \cdots, u_n]^T$$

$$\mathbf{o} = [o_1, o_2, \cdots, o_m]^T$$

Given an input pattern **x**, the iteration starts from the second layer:

$$o_j(1) = \varphi(\sum_{i=1}^{d} w_{ij} x_i)$$

In the second iteration, the first layer gives:

$$u_j(2) = \varphi[\sum_{i=1}^{m} w_{ji} o_i(1)]$$

And the second layer gives:

$$o_j(2) = \varphi[\sum_{i=1}^{d} w_{ij} u_i(2)]$$

The iteration is continued until no updating occurs to all the neurons in the first and the second layers.

The weights can be computed using the following formula:

$$w_{ij} = \sum_{k=1}^{N} x_{ki} y_{kj}$$

In matrix form:

$$\mathbf{W} = \sum_{i=1}^{N} \mathbf{x}_i \mathbf{y}_i^T$$

Example 1:

Consider two pairs of patterns

$$\mathbf{x}_1 = [1 \quad 1 \quad -1]^T \qquad \mathbf{y}_1 = [-1 \quad 1 \quad -1 \quad 1]^T$$

$$\mathbf{x}_2 = [1 \quad -1 \quad 1]^T \qquad \mathbf{y}_2 = [1 \quad 1 \quad 1 \quad 1]^T$$

Then the matrix $\mathbf{W}$ is computed as follow:

$$\mathbf{W} = \sum_{i=1}^{2} \mathbf{x}_i \mathbf{y}_i^T = \begin{bmatrix} 0 & 2 & 0 & 2 \\ -2 & 0 & -2 & 0 \\ 2 & 0 & 2 & 0 \end{bmatrix}$$

If we have an input pattern $\mathbf{x}$:

$$\mathbf{x} = [+1, +1, -1]^T$$

In the first iteration, the output of feedback layer is:

$$\mathbf{o}(1) = \varphi(\mathbf{W}^T \mathbf{x}) = \varphi([-4 \quad 2 \quad -4 \quad 2]^T) = [-1 \quad 1 \quad -1 \quad 1]^T$$

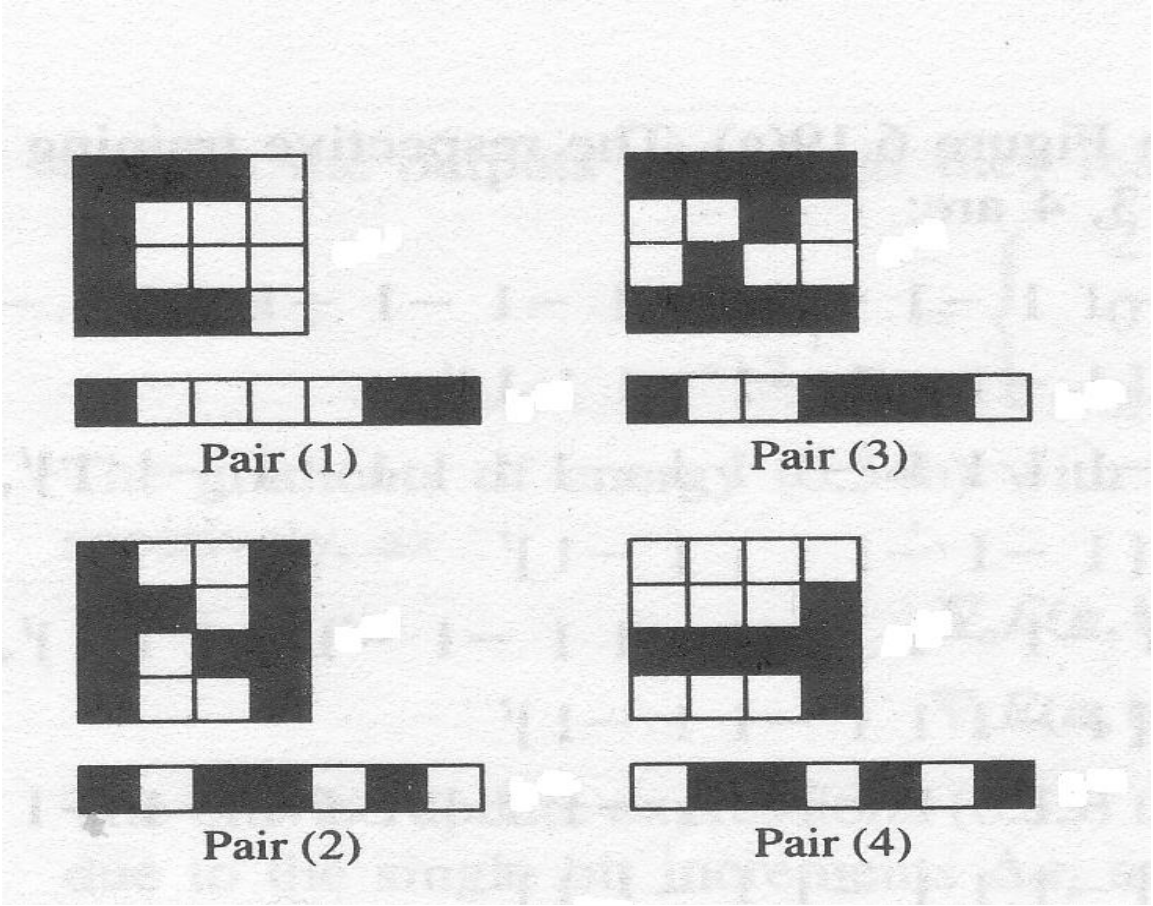
In the second iteration:

$$\mathbf{u}(2) = \varphi[\mathbf{W}\mathbf{o}(1)] = \varphi([4 \quad 4 \quad -4]^T) = [1 \quad 1 \quad -1]^T$$

$$\mathbf{o}(2) = \varphi[\mathbf{W}^T \mathbf{u}(2)] = \varphi([-4 \quad 2 \quad -4 \quad 2]^T) = [-1 \quad 1 \quad -1 \quad 1]^T$$

Example 2:

The association is from a 16-pixel map of letter characters to a 7-bit binary vector. The objective is to design a BAM to store the associations.



The images can be represented by the following vectors (assume black pixel is denoted by 1, white pixel -1)

$$\mathbf{x}_1 = [1,1,1,-1,1,-1,-1,-1,1,-1,-1,-1,1,1,1,-1]^T$$

$$\mathbf{y}_1 = [1,-1,-1,-1,-1,1,1]^T$$

$$\mathbf{x}_2 = [1,-1,-1,1,1,1,-1,1,1,-1,1,1,1,-1,-1,1]^T$$

$$\mathbf{y}_2 = [1,-1,1,1,-1,1,-1]^T$$

$$\mathbf{x}_3 = [1,1,1,1,-1,-1,1,-1,-1,1,-1,-1,1,1,1,1]^T$$

$$\mathbf{y}_3 = [1,-1,-1,1,1,1,-1]^T$$

$$\mathbf{x}_4 = [-1,-1,-1,-1,-1,-1,-1,1,1,1,1,1,-1,-1,-1,1]^T$$

$$\mathbf{y}_4 = [-1,1,1,-1,1,-1,1]^T$$

Because the input and output are vectors with dimension 16 and 7 respectively, the structure of the BAM is 16×7. The weight matrix is obtained as:

$$\mathbf{W} = \begin{bmatrix} 4 & -4 & -2 & 2 & -2 & 4 & -2 \\ 2 & -2 & -4 & 0 & 0 & 2 & 0 \\ 2 & -2 & -4 & 0 & 0 & 2 & 0 \\ 2 & -2 & 0 & 4 & 0 & 2 & -4 \\ 2 & -2 & 0 & 0 & -4 & 2 & 0 \\ 0 & 0 & 2 & 2 & -2 & 0 & -2 \\ 0 & 0 & -2 & 2 & 2 & 0 & -2 \\ -2 & 2 & 4 & 0 & 0 & -2 & 0 \\ 0 & 0 & 2 & -2 & -2 & 0 & 2 \\ -2 & 2 & 0 & 0 & 4 & -2 & 0 \\ -2 & 2 & 4 & 0 & 0 & -2 & 0 \\ -2 & 2 & 4 & 0 & 0 & -2 & 0 \\ 4 & -4 & -2 & 2 & -2 & 4 & -2 \\ 2 & -2 & -4 & 0 & 0 & 2 & 0 \\ 2 & -2 & -4 & 0 & 0 & 2 & 0 \\ 0 & 0 & 2 & 2 & 2 & 0 & -2 \end{bmatrix}$$

Consider a distorted $\mathbf{x}_1$:

$$\mathbf{x} = [1,1,1,1,1,-1,-1,-1,1,-1,-1,-1,1,1,1,-1]^T$$

Then:

$$\mathbf{o}(1) = \varphi(\mathbf{W}^T\mathbf{x}) = [1,-1,-1,-1,-1,1,1]^T$$

However, the error tolerating ability of the network is not unlimited.

Consider another distorted $\mathbf{x}_1$:

$$\mathbf{x} = [-1,-1,-1,1,-1,1,-1,1,1,-1,1,1,-1,-1,1,1]^T$$

$$\mathbf{o}(1) = \phi(\mathbf{W}^T\mathbf{x}) = [-1,1,1,0,0,-1,1]^T$$

$$\mathbf{u}(2) = \phi[\mathbf{W}\mathbf{o}(1)] = [-1,-1,-1,-1,-1,1,-1,1,1,1,1,1,-1,-1,-1,1]^T$$

$$\mathbf{o}(2) = \phi[\mathbf{W}^T\mathbf{u}(2)] = [-1,1,1,-1,1,-1,1]^T$$

$$\mathbf{u}(3) = \varphi[\mathbf{W}\mathbf{o}(2)] = [-1,-1,-1,-1,-1,1,-1,1,1,1,1,1,-1,-1,-1,1]^T$$

$$\mathbf{o}(3) = \varphi[\mathbf{W}^T\mathbf{u}(3)] = [-1,1,1,-1,1,-1,1]^T$$

The distorted $\mathbf{x}_1$ converges to $\mathbf{y}_4$. This is because the distorted $\mathbf{x}_1$ is more similar to $\mathbf{x}_4$ than to $\mathbf{x}_1$