Hopfield Network

Nguyen Ngoc Thao

Department of Computer Science, FIT
University of Science, VNU-HCM

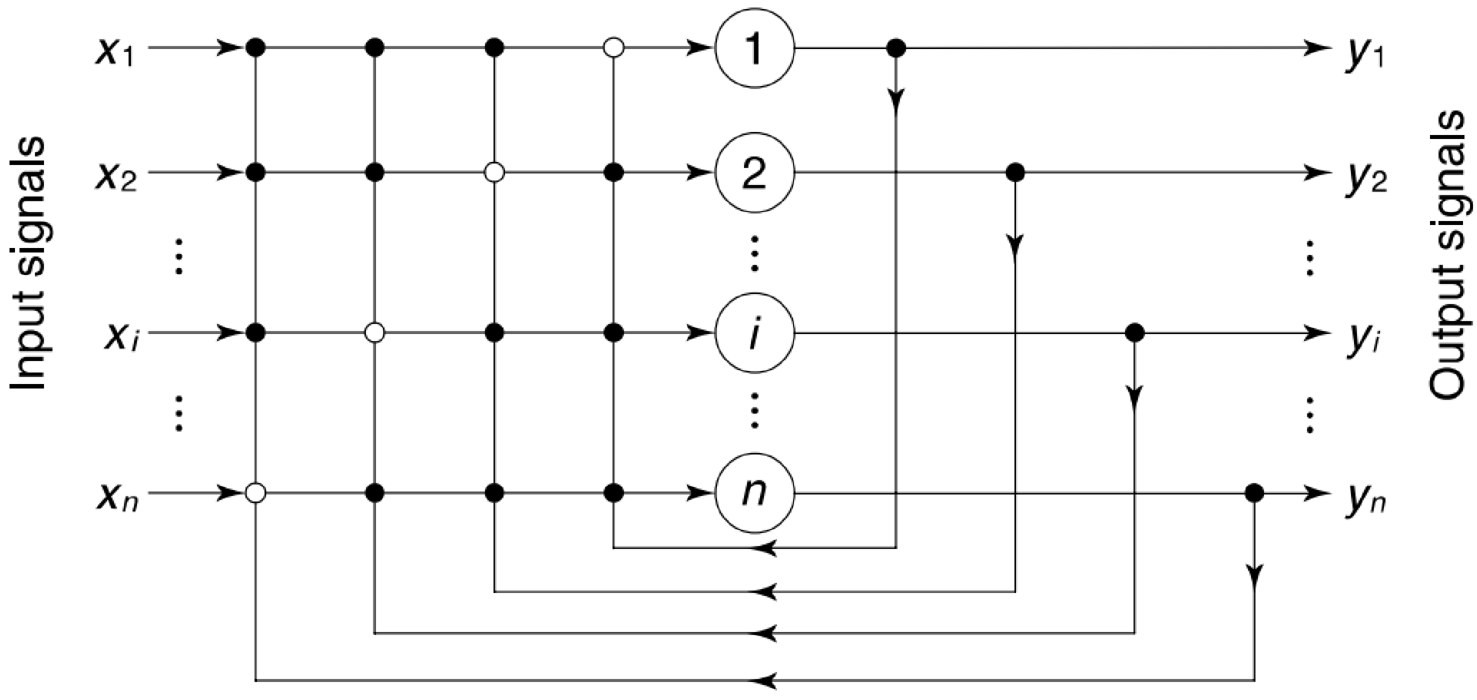
Content outline

- Hopfield network
- Bidirectional associative memory (BAM)

Hopfield network

Hopfield network (Hopfield, 1982)

- A stable network that usually uses McCulloch-Pitts neuron and the sign activation function
- No self-feedback

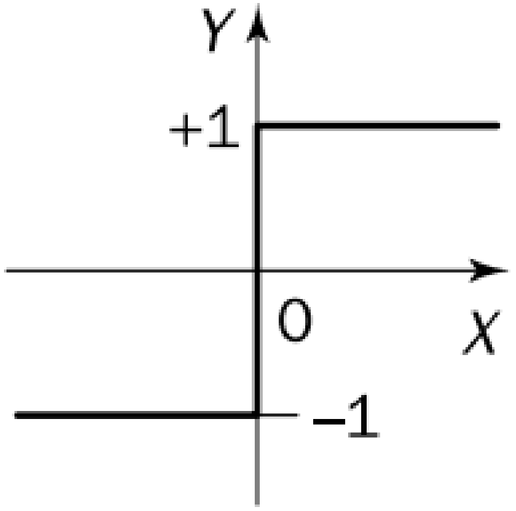


A single-layer n-neuron Hopfield network

Hopfield network: Activation function

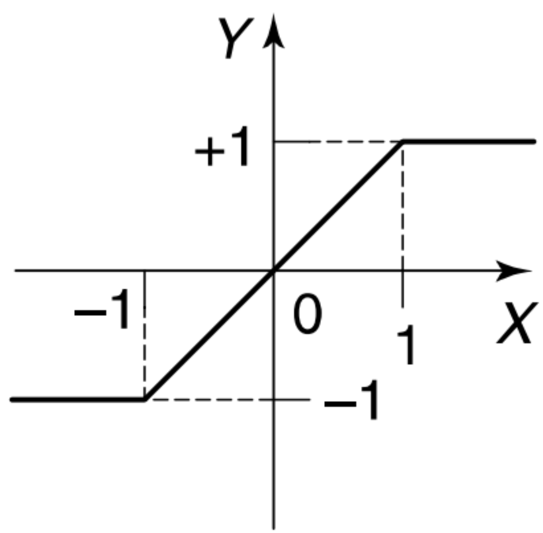
- The sign activation function

$$Y^{sign} = \begin{cases} +1 & if\ X > 0 \\ -1 & if\ X < 0 \\ X & if\ X = 0 \end{cases}$$



- This may be replaced with a saturated linear function

$$Y^{satlin} = \begin{cases} X & if\ -1 < X < 1 \\ +1 & if\ X \geq 1 \\ -1 & if\ X \leq -1 \end{cases}$$

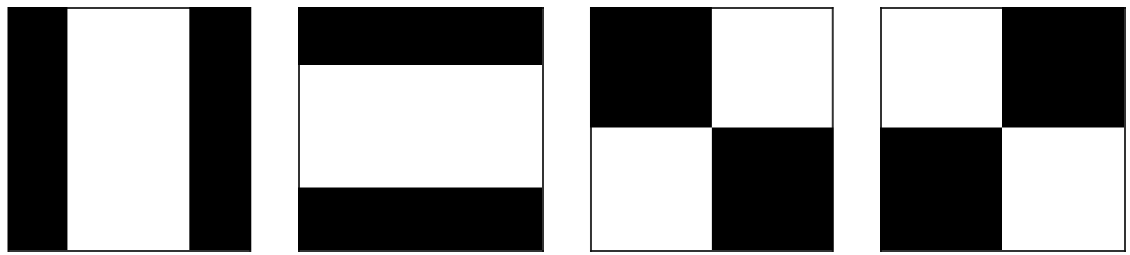


How do Hopfield networks learn?

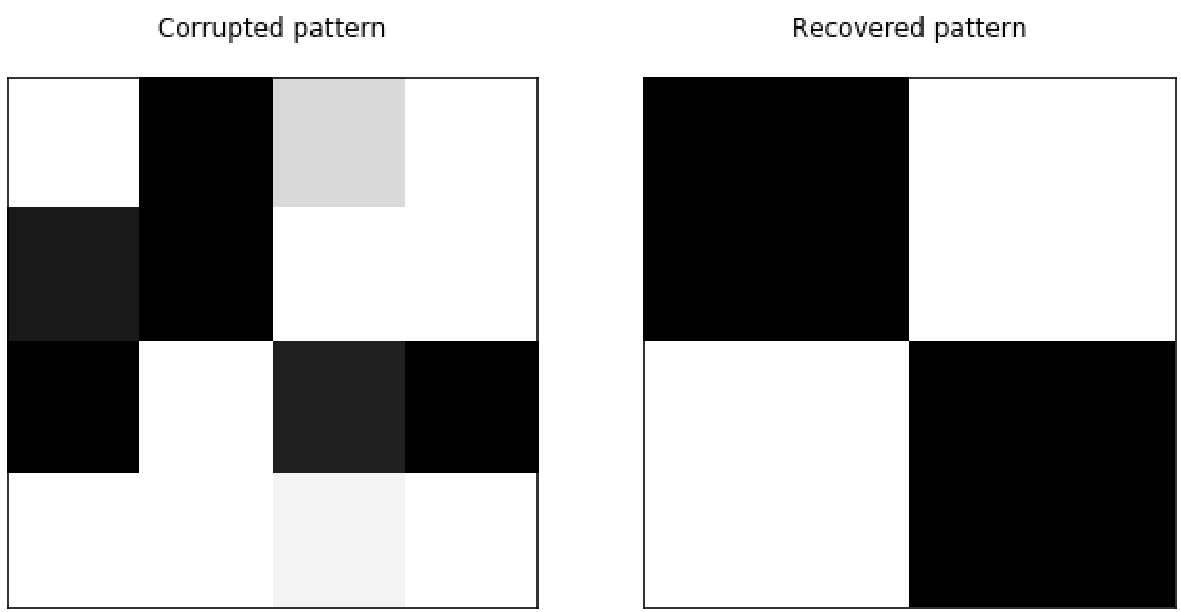
- Hopfield nets serve as content-addressable (associative) memory systems with binary threshold nodes (-1 or 1).
- They are guaranteed to converge to a local minimum.
  - The state of a node becomes fixed after a certain number of updates.
- However, maybe to a false pattern (wrong local minimum) rather than a stored pattern (expected local minimum)

Hopfield network: Another example

- The patterns to be remembered are

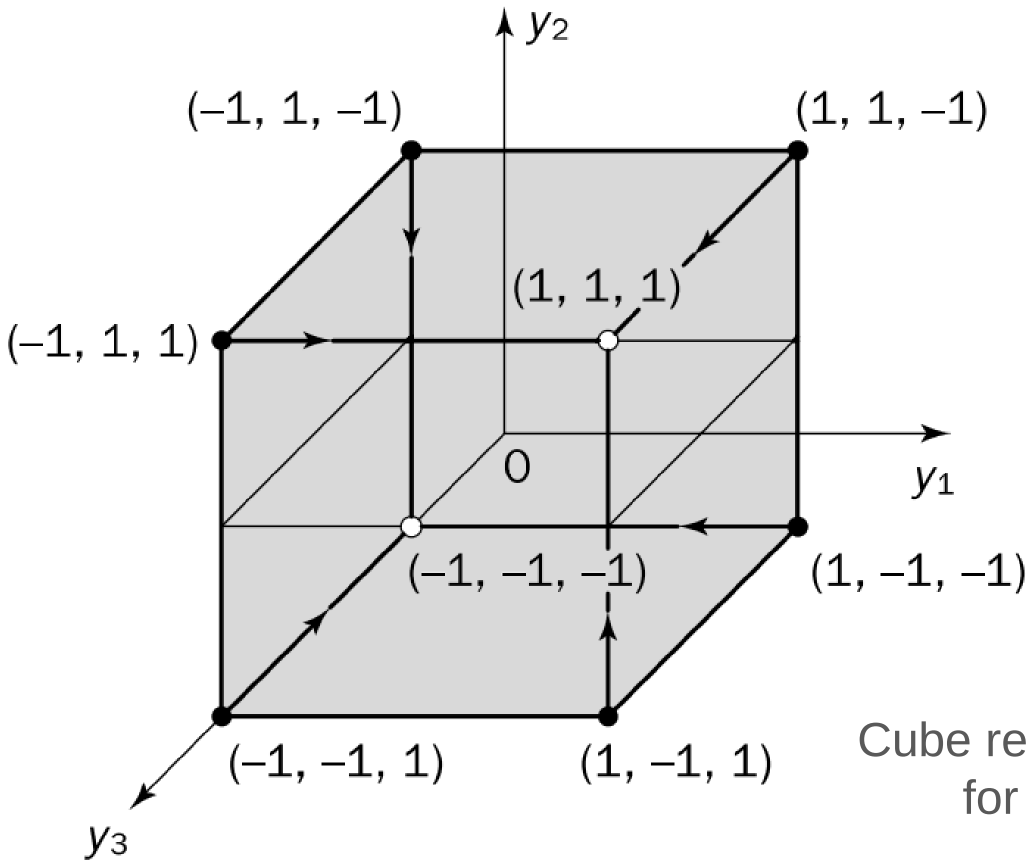


- A corrupted pattern is recovered to the closest memorized pattern.



How do Hopfield networks learn?

- A network of n neurons generally has 2^n possible states, i.e., it is associated with an n-dimensional hypercube.



Cube representation of the possible states for the three-neuron Hopfield network

How do Hopfield networks learn?

- The current state of the network is determined by the current outputs of all neurons, called the state vector.

$$\mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

- Synaptic weights between neurons are represented by

$$\mathbf{W} = \sum_{m=1}^{M} \mathbf{Y}_{\boldsymbol{m}} \mathbf{Y}_{\boldsymbol{m}}^{\boldsymbol{T}} - M\mathbf{I}$$

where $\boldsymbol{M}$ is the number of states to be memorized by the network, $\mathbf{Y}_{\boldsymbol{m}}$ is a n-dimensional binary vector, $\mathbf{I}$ is $n \times n$ identity matrix.

How do Hopfield networks learn?

- Suppose that the network is required to memorize two opposite states, $\mathbf{Y_1} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ and $\mathbf{Y_2} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$

- Thus, the weight matrix is determined as

$$\mathbf{W} = \mathbf{Y_1 Y_1^{\mathit{T}}} + \mathbf{Y_2 Y_2^{\mathit{T}}} - \mathbf{2I}$$

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

How to test Hopfield network?

- Consider the input test vectors, $\mathbf{X_1}$ and $\mathbf{X_2}$, which are equal to the output vectors, $\mathbf{Y_1}$ and $\mathbf{Y_2}$, respectively.
- The actual output vector is $\mathbf{Y}_m = sign(\mathbf{W}\mathbf{X}_m - \boldsymbol{\theta})$

  where $\boldsymbol{\theta}$ is the threshold matrix and $\boldsymbol{m} = 1,2, \dots, M$
- This example assumes all thresholds to be zero.
- Thus,

$$\mathbf{Y}_1 = sign\left\{ \begin{bmatrix} 0 & 2 & 2 \\ 2 & 0 & 2 \\ 2 & 2 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right\} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$\mathbf{Y}_2 = sign\left\{ \begin{bmatrix} 0 & 2 & 2 \\ 2 & 0 & 2 \\ 2 & 2 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right\} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$$

- Both $(1, 1, 1)$ and $(-1, -1, -1)$ states are said to be stable.

How to test Hopfield network?

- The remaining six states are all unstable.

Possible state	Iteration	Inputs x_1	x_2	x_3	Outputs y_1	y_2	y_3	Fundamental memory
1 1 1	0	1	1	1	1	1	1	1 1 1
−1 1 1	0	−1	1	1	1	1	1	
	1	1	1	1	1	1	1	1 1 1
1 −1 1	0	1	−1	1	1	1	1	
	1	1	1	1	1	1	1	1 1 1
1 1 −1	0	1	1	−1	1	1	1	
	1	1	1	1	1	1	1	1 1 1
−1 −1 −1	0	−1	−1	−1	−1	−1	−1	−1 −1 −1
−1 −1 1	0	−1	−1	1	−1	−1	−1	
	1	−1	−1	−1	−1	−1	−1	−1 −1 −1
−1 1 −1	0	−1	1	−1	−1	−1	−1	
	1	−1	−1	−1	−1	−1	−1	−1 −1 −1
1 −1 −1	0	1	−1	−1	−1	−1	−1	
	1	−1	−1	−1	−1	−1	−1	−1 −1 −1

Stable vs. Unstable states

- Stable states (or fundamental memories) can attract states that are close to them.
- Each unstable state represents a single error, compared to the corresponding fundamental memory.
- The Hopfield network acts as an error correction network.

Hopfield network learning rule

- Step 1: Storage
  - The n-neuron Hopfield network is required to store a set of M fundamental memories, $\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_M$.
  - The synaptic weight from neuron i to neuron j is calculated as

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

where $\boldsymbol{y_{m,i}}$ and $\boldsymbol{y_{m,j}}$ are the i^{th} and j^{th} elements of $\mathbf{Y}_M$, respectively

Or, in matrix form, $\mathbf{W} = \sum_{m=1}^{M} \mathbf{Y}_{\boldsymbol{m}} \mathbf{Y}_{\boldsymbol{m}}^{\boldsymbol{T}} - M\mathbf{I}$

Hopfield network learning rule

- Step 1: Storage (cont.)
  - Fundamental memories can be stored if the weight matrix is symmetric with zeros in main diagonal (Cohen and Grossberg, 1983).
  - The weights remain fixed after calculation.

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

Hopfield network learning rule

- Step 2: Testing
  - The network must recall any fundamental memory $\mathbf{Y}_M$ when presented with it as an input.
  - That is, given $\boldsymbol{X_m} = \boldsymbol{Y_m}\ (m = 1, 2, \ldots, M)$, $\boldsymbol{Y_m} = \mathbf{sign}(\boldsymbol{W}\boldsymbol{X_m} - \boldsymbol{\theta})$

    where $\boldsymbol{y_{m,i}}$ is the i^{th} element of the actual output vector $\mathbf{Y}_M$, and $\boldsymbol{x_{m,j}}$ is the j^{th} element of the input vector $\boldsymbol{X}_M$.
  - If all fundamental memories are recalled perfectly, proceed to the next step.

Hopfield network learning rule

- Step 3: Retrieval
  - Present an unknown n-dimensional vector, $\mathbf{X}$, to the network and retrieve a stable state.
    - $\mathbf{X}$ typically represents a corrupted or incomplete version of the fundamental memory: $\mathbf{X} \neq \mathbf{Y}_m,\ m = 1, 2, \dots, M$
  - (a) Initialize the retrieval algorithm by setting $\boldsymbol{X}(\mathbf{0}) = \boldsymbol{X}$ and calculate the initial state vector at iteration $\boldsymbol{p} = \mathbf{0}$

$$\boldsymbol{Y}(\mathbf{0}) = \mathbf{sign}(\boldsymbol{W}\boldsymbol{X}(\mathbf{0}) - \boldsymbol{\theta})$$

    b) Update the state vector

$$\boldsymbol{Y}(\boldsymbol{p} + \mathbf{1}) = \mathbf{sign}(\boldsymbol{W}\boldsymbol{X}(\boldsymbol{p}) - \boldsymbol{\theta})$$

  - Neurons for updating are selected randomly and one at a time.
  - Repeat the iteration until the state vector becomes unchanged

$$\boldsymbol{Y}(\boldsymbol{p} + \mathbf{1}) = \mathbf{sign}(\boldsymbol{W}\boldsymbol{Y}(\boldsymbol{p}) - \boldsymbol{\theta})$$

Autoassociative memory

- Hopfield network acts as autoassociative memory.
- It can retrieve a piece of data upon presentation of only partial information from that piece of data.
- For example,
  - The sentence fragments presented below are sufficient for most humans to recall the missing information.

    "To be or not to be, that is _____."

    "I came, I saw, _____."
  - Many readers will realize the missing information is in fact:

    "To be or not to be, that is the question."

    "I came, I saw, I conquered."

About the Hopfield network

- Hopfield network will always converge to a stable state if the retrieval is done asynchronously (Haykin, 1999).
- This stable state does not necessarily represent one of the fundamental memories or the closest fundamental memory.
  - E.g., the following network produced for $\mathbf{X}$ a pattern recalling $\mathbf{X}_3$ instead of $\mathbf{X}_1$

$$\mathbf{X}_1 = (+1,+1,+1,+1,+1)$$
$$\mathbf{X}_2 = (+1,-1,+1,-1,+1)$$
$$\mathbf{X}_3 = (-1,+1,-1,+1,-1)$$

$$\boxed{\mathbf{X} = (+1,+1,-1,+1,+1)}$$

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

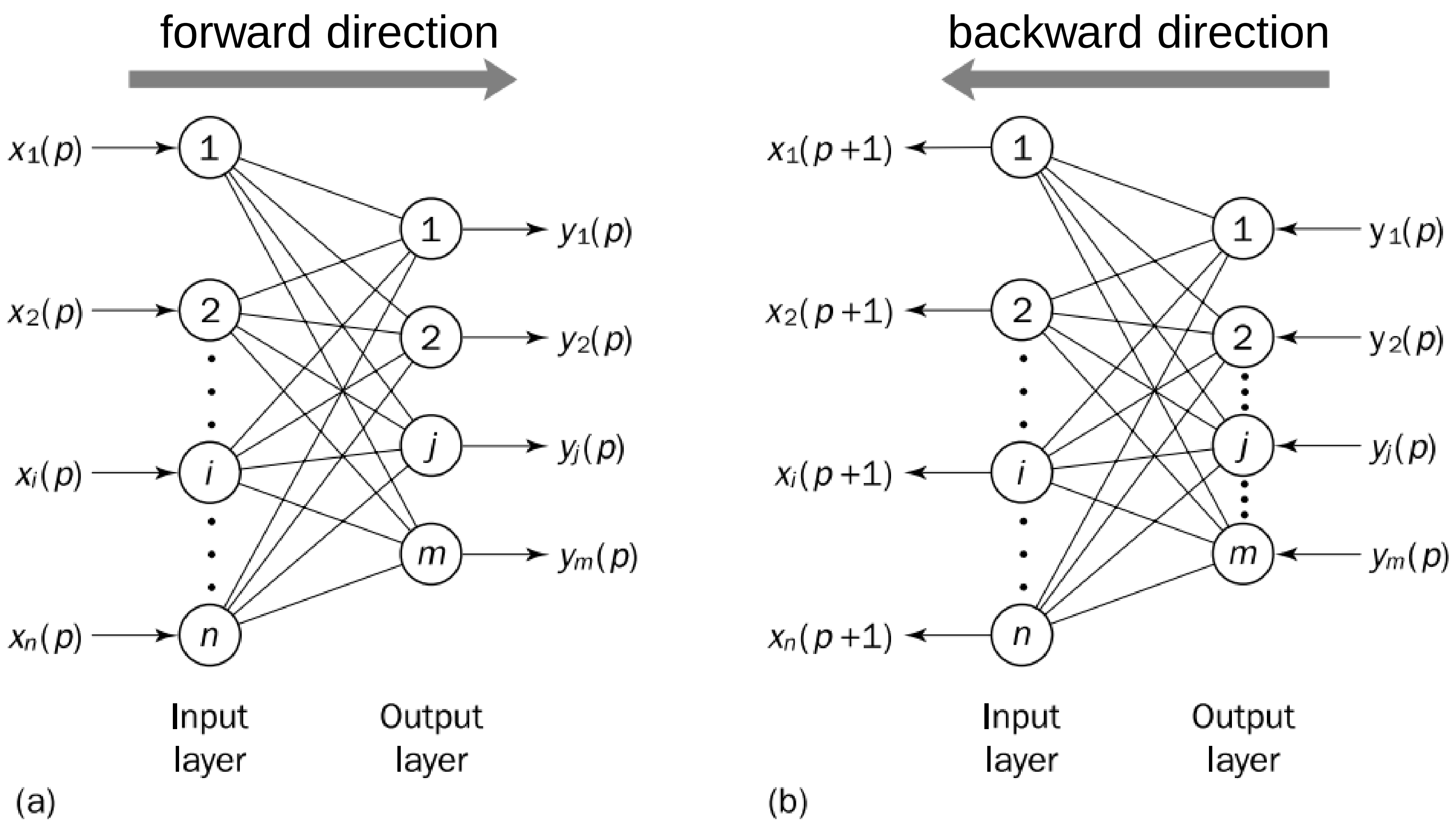
About the Hopfield network

- Storage capacity: the largest number of fundamental memories that can be stored and retrieved correctly.
- (Hopfield, 1982): the storage capacity is experimentally limited by $M_{max} = 0.15n$.
- (Amit, 1989): the storage capacity for most fundamental memories to be retrieved perfectly is $M_{max} = \frac{n}{2 \ln n}$, and to retrieve **all** fundamental memories perfectly, $M_{max} = \frac{n}{4 \ln n}$
- Major limitation: the storage capacity must be kept rather small for the fundamental memories to be retrievable

Bidirectional associative memory (BAM)

BAM (Kosko, 1987, 1988)

- The BAM network acts as heteroassociative memory.
- Patterns from one set, A, are associated to patterns from another set, B, and vice versa.



How does BAM work?

- Pattern pairs are stored.
- When the n-dimensional vector $\mathbf{X} \in A$ is presented as input, the BAM recalls the m-dimensional vector $\mathbf{Y} \in B$, and when $\mathbf{Y}$ is presented as input, the BAM recalls $\mathbf{X}$**.**
- This process is repeated until input and output vectors become unchanged.

BAM learning rule

- Step 1: Storage
  - The BAM is required to store M pairs of patterns.
  - For example,

$$\text{Set } A: \quad \mathbf{X}_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad \mathbf{X}_2 = \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \\ -1 \\ -1 \end{bmatrix} \quad \mathbf{X}_3 = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \\ 1 \\ 1 \end{bmatrix} \quad \mathbf{X}_4 = \begin{bmatrix} -1 \\ -1 \\ 1 \\ 1 \\ -1 \\ -1 \end{bmatrix}$$

$$\text{Set } B: \quad \mathbf{Y}_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad \mathbf{Y}_2 = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} \quad \mathbf{Y}_3 = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} \quad \mathbf{Y}_4 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}$$

BAM learning rule

- Step 1: Storage (cont.)
  - The weight matrix is determined as $\mathbf{W} = \sum_{m=1}^{M} \mathbf{X}_m \mathbf{Y}_\mathbf{m}^\mathbf{T}$

    where M is the number of pattern pairs to be stored in the BAM
  - For example,

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

$$+ \begin{bmatrix} -1 \\ -1 \\ 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} \begin{bmatrix} -1 & 1 & -1 \end{bmatrix} = \begin{bmatrix} 4 & 0 & 4 \\ 4 & 0 & 4 \\ 0 & 4 & 0 \\ 0 & 4 & 0 \\ 4 & 0 & 4 \\ 4 & 0 & 4 \end{bmatrix}$$

BAM learning rule

- **Step 2: Testing**
  - Confirm that the BAM is able to recall $\mathbf{Y}_m$ when presented with $\mathbf{X}_m$.
  - That is,

$$\mathbf{Y}_m = \mathbf{sign}(\mathbf{W}^T\mathbf{X}_m), \text{ where } m = 1,2, \dots, M$$

  - For example,

$$\mathbf{Y}_1 = sign\left(\mathbf{W}^T\mathbf{X}_1\right) = sign\left\{\begin{bmatrix} 4 & 4 & 0 & 0 & 4 & 4 \\ 0 & 0 & 4 & 4 & 0 & 0 \\ 4 & 4 & 0 & 0 & 4 & 4 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}\right\} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

BAM learning rule

- Step 2: Testing (cont.)
  - Then confirm that the BAM recalls $\mathbf{X}_m$ when presented with $\mathbf{Y}_m$.
  - That is,

$$\mathbf{X}_m = \mathbf{sign}(\mathbf{W}\mathbf{Y}_m), \text{ where } m = 1,2,\ldots,M$$

  - For example,

$$\mathbf{X}_3 = sign\,(\mathbf{W}\,\mathbf{Y}_3) = sign\left\{\begin{bmatrix} 4 & 0 & 4 \\ 4 & 0 & 4 \\ 0 & 4 & 0 \\ 0 & 4 & 0 \\ 4 & 0 & 4 \\ 4 & 0 & 4 \end{bmatrix}\begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}\right\} = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \\ 1 \\ 1 \end{bmatrix}$$

  - If all pairs are recalled perfectly, proceed to the next step

BAM learning rule

- Step 3: Retrieval
  - Present an unknown vector $\mathbf{X}$, which is a corrupted or incomplete version of a pattern from A (or B) stored in the BAM.
  - That is, $\mathbf{X} \neq \boldsymbol{X}_m,\ m = 1, 2, \ldots, M$
  - (a) Initialize the BAM retrieval algorithm by setting $\mathbf{X}(\mathbf{0}) = \mathbf{X}, \boldsymbol{p} = \mathbf{0}$

    and calculate the BAM output at iteration $\boldsymbol{p}$

$$\mathbf{Y}(\boldsymbol{p}) = \mathbf{sign}[\mathbf{W}^{\boldsymbol{T}}\mathbf{X}(\boldsymbol{p})]$$

  - (b) Update the input vector $\mathbf{X}(\boldsymbol{p})$

$$\mathbf{X}(\boldsymbol{p} + \mathbf{1}) = \mathbf{sign}[\mathbf{W}\mathbf{Y}(\boldsymbol{p})]$$

    and repeat the iteration until equilibrium.

    The input and output patterns represent an associated pair

About the BAM network

- Hopfield network is a BAM special case when the weight matrix is square and symmetric.
- The maximum number of associations to be stored should not exceed the number of neurons in the smaller layer.
- Unconditionally stable (Kosko, 1992): any set of associations can be learned without risk of instability


THE
END