SUPERVISED HEBBIAN LEARNING

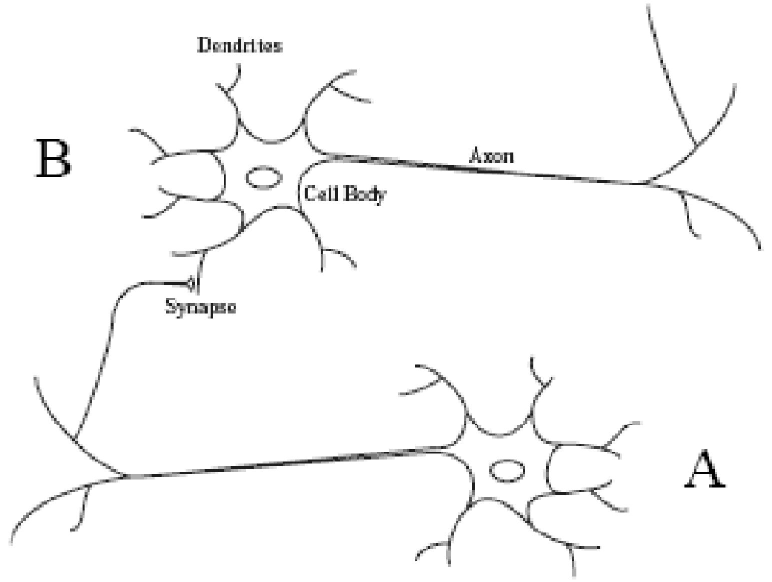
CSC 302 1.5 Neural Networks

Entrance

- One of the first neural network learning rules [1949].
- Proposed by Donald Hebb [1949] as a possible mechanism for synaptic modification in the brain.
- Describe how Hebb rule can be used to train neural networks for pattern recognition.

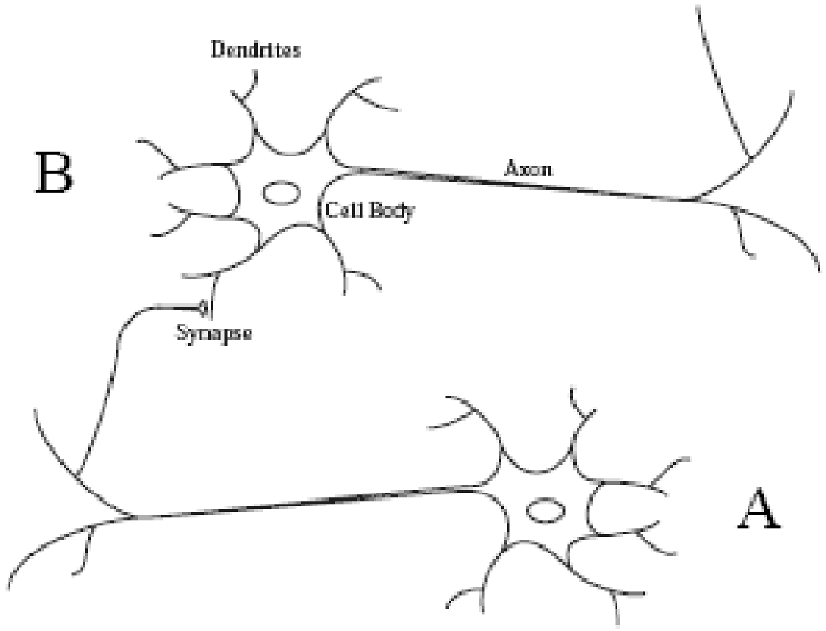
Hebb’s Postulate (1)

- When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased.



Hebb’s Postulate (2)

- The simultaneous excitation of two neurons result in a strengthening of the connection between them.

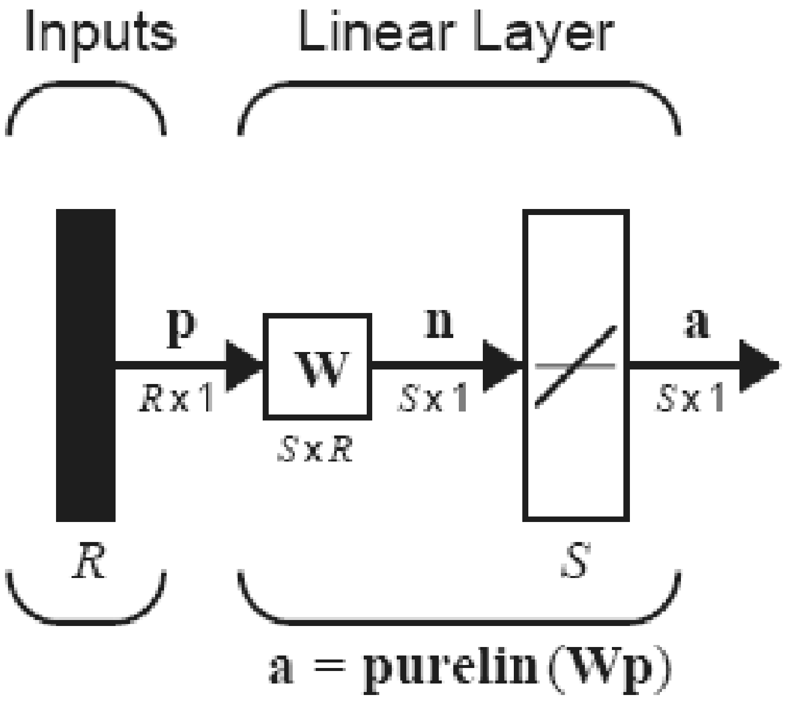


Hebbian Learning Rule

- Connections between two neurons might be strengthened if the neurons fire simultaneously.
- How much the weight of the connection between two neurons should be increased or decreased in proportion to the product of their activation.
- Works well as long as all the input patterns are orthogonal or uncorrelated.

Two vectors **u** and **v** whose dot product is **u·v = 0** (i.e., the vectors are perpendicular) are said to be orthogonal.

Linear Associator



$$\mathbf{a} = \mathbf{W}\mathbf{p}$$

Simplest architecture in which the Hebb's rule can be used.

$$a_i = \sum_{j=1}^{R} w_{ij} p_j$$

Training Set:

$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \ldots, \{\mathbf{p}_Q, \mathbf{t}_Q\}$$

w_{ij} – connection (synapse) between input p_j and output of the *i*th neuron a_i

Associative Memory

- Task – Learn Q pairs of prototype input/output vectors:

$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \dots, \{\mathbf{p}_Q, \mathbf{t}_Q\}$$

- In addition, if the input is changed slightly, then the output should only be changed slightly.

$$\mathbf{p} = \mathbf{p_q} + \delta \Rightarrow \mathbf{a} = \mathbf{t_q} + \varepsilon$$

- E.g. Linear associator

The Hebb Rule (1)

- How can we interpret Hebb’s postulate mathematically?
- If two neurons of a connection are activated simultaneously, the weight of the connection will increase.

$$w_{ij}^{new} = w_{ij}^{old} + \alpha \Delta w_{ij}$$

- w_{ij} increases if a positive p_j produces a positive a_i.

The Hebb Rule (2)

$$w_{ij}^{new} = w_{ij}^{old} + \alpha\ f_i(a_{iq})g_j(p_{jq})$$

Pre-synaptic Signal

Post-synaptic Signal

- a_{iq} – i^{th} element of the network output when the q^{th} input vector $\mathbf{p_q}$ is presented to the network.
- p_{jq} – j^{th} element of the q^{th} input vector $\mathbf{p_q}$.
- α - learning rate (positive constant).
- Change in the weight w_{ij} is proportional to a product of functions of the activities on either side of the synapse.

The Hebb Rule (3)

Simplified form:

$$w_{ij}^{new} = w_{ij}^{old} + \alpha \, a_{iq} p_{jq}$$

Increases the weight when both $\mathtt{p_j}$ and $\mathtt{a_i}$ are +ve as well as both $\mathtt{p_j}$ and $\mathtt{a_i}$ are –ve.

Decreases the weight when $\mathtt{p_j}$ and $\mathtt{a_i}$ have opposite sign.

Unsupervised learning

Supervised form:

$$w_{ij}^{new} = w_{ij}^{old} + \alpha \, t_{iq} p_{jq}$$

Replaces the actual output ($\mathtt{a_i}$) by the target output ($\mathtt{t_i}$).

Matrix form:

$$\mathbf{W}^{\mathbf{new}} = \mathbf{W}^{\mathbf{old}} + \alpha \, \mathbf{t_q p_q^T}$$

$$\mathbf{W}^{\mathbf{new}} = \mathbf{W}^{\mathbf{old}} + \mathbf{t_q p_q^T}$$

For simplicity, set α = 1

Batch Operation

$$\mathbf{W} = \mathbf{t}_1\mathbf{p}_1^T + \mathbf{t}_2\mathbf{p}_2^T + \cdots + \mathbf{t}_Q\mathbf{p}_Q^T = \sum_{q=1}^{Q} \mathbf{t}_q\mathbf{p}_q^T$$

(Zero Initial Weights)

Matrix Form:

$$\mathbf{W} = \begin{bmatrix} \mathbf{t}_1 & \mathbf{t}_2 & \dots & \mathbf{t}_Q \end{bmatrix} \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \\ \vdots \\ \mathbf{p}_Q^T \end{bmatrix} = \mathbf{T}\mathbf{P}^T$$

Input vectors

$$\mathbf{P} = \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 & \cdots & \mathbf{p}_Q \end{bmatrix}$$

$$\mathbf{T} = \begin{bmatrix} \mathbf{t}_1 & \mathbf{t}_2 & \dots & \mathbf{t}_Q \end{bmatrix}$$

Target vectors

Performance Analysis

$$\mathbf{a} = \mathbf{W}\mathbf{p}_k = \left(\sum_{q=1}^{Q} \mathbf{t}_q \mathbf{p}_q^T\right)\mathbf{p}_k = \sum_{q=1}^{Q} \mathbf{t}_q (\mathbf{p}_q^T \mathbf{p}_k)$$

Case 1, input patterns are orthonormal (orthogonal and unit length)

$$\begin{aligned} (\mathbf{p}_q^T \mathbf{p}_k) &= 1 \qquad q = k \\ &= 0 \qquad q \neq k \end{aligned}$$

Therefore the network output equals the target:

$$\mathbf{a} = \mathbf{W}\mathbf{p}_k = \mathbf{t}_k$$

Case 2, input patterns are normalized (unit length), but not orthogonal

$$\mathbf{a} = \mathbf{W}\mathbf{p}_k = \mathbf{t}_k + \underbrace{\boxed{\sum_{q \neq k} \mathbf{t}_q (\mathbf{p}_q^T \mathbf{p}_k)}}_{\text{Error}}$$

Example (Orthogonal) (1)

$$\left\{p_1 = \begin{bmatrix} 0.5 \\ -0.5 \\ 0.5 \\ -0.5 \end{bmatrix}, \quad t_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}\right\} \qquad \left\{p_2 = \begin{bmatrix} 0.5 \\ 0.5 \\ -0.5 \\ -0.5 \end{bmatrix}, \quad t_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}\right\}$$

Ortho-normal

$$\mathbf{P} = [\mathbf{p}_1 \ \mathbf{p}_2] = \begin{bmatrix} 0.5 & 0.5 \\ -0.5 & 0.5 \\ 0.5 & -0.5 \\ -0.5 & -0.5 \end{bmatrix} \qquad \mathbf{T} = [\mathbf{t}_1 \ \mathbf{t}_2] = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$$

The weight matrix (Hebb rule):

$$\mathbf{W} = \mathbf{T}\mathbf{P}^{\mathbf{T}} = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 0.5 & -0.5 & 0.5 & -0.5 \\ 0.5 & 0.5 & -0.5 & -0.5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \end{bmatrix}$$

Example (Orthogonal) (2)

$$\mathbf{Wp_1} = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} 0.5 \\ -0.5 \\ 0.5 \\ -0.5 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

$$\mathbf{Wp}_2 = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0.5 \\ -0.5 \\ -0.5 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Success! outputs and targets are equal

Example (Apple/Banana)

Banana Apple Normalized Prototype Patterns

$$\mathbf{p}_1 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} \quad \mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} \quad \left\{ \mathbf{p}_1 = \begin{bmatrix} -0.5774 \\ 0.5774 \\ -0.5774 \end{bmatrix}, \mathbf{t}_1 = \begin{bmatrix} -1 \end{bmatrix} \right\} \quad \left\{ \mathbf{p}_2 = \begin{bmatrix} 0.5774 \\ 0.5774 \\ -0.5774 \end{bmatrix}, \mathbf{t}_2 = \begin{bmatrix} 1 \end{bmatrix} \right\}$$

Weight Matrix (Hebb Rule):

$$\mathbf{W} = \mathbf{T}\mathbf{P}^T = \begin{bmatrix} -1 & 1 \end{bmatrix} \begin{bmatrix} -0.5774 & 0.5774 & -0.5774 \\ 0.5774 & 0.5774 & -0.5774 \end{bmatrix} = \begin{bmatrix} 1.1548 & 0 & 0 \end{bmatrix}$$

Tests:

Banana $$\mathbf{W}\mathbf{p}_1 = \begin{bmatrix} 1.1548 & 0 & 0 \end{bmatrix} \begin{bmatrix} -0.5774 \\ 0.5774 \\ -0.5774 \end{bmatrix} = \begin{bmatrix} -0.6668 \end{bmatrix}$$

Apple $$\mathbf{W}\mathbf{p}_2 = \begin{bmatrix} 0 & 1.1548 & 0 \end{bmatrix} \begin{bmatrix} 0.5774 \\ 0.5774 \\ -0.5774 \end{bmatrix} = \begin{bmatrix} 0.6668 \end{bmatrix}$$

Failed! Outputs and targets are not equal.

Pseudoinverse Rule (1)

- The Hebb rule produces errors when the input vectors are not orthogonal.
- Several procedures to reduce these errors.
- E.g. Pseudo-inverse rule

Performance Index: $\mathbf{W}\mathbf{p}_q = \mathbf{t}_q \qquad q = 1, 2, \dots, Q$

$$F(\mathbf{W}) = \sum_{q=1}^{Q} \|\mathbf{t}_q - \mathbf{W}\mathbf{p}_q\|^2$$

Goal – selecting a weight matrix to minimize *F*(**W**)

Pseudoinverse Rule (2)

Matrix Form: $\mathbf{W}\mathbf{P} = \mathbf{T}$ $\longleftarrow$ Expectation

$$\mathbf{T} = \begin{bmatrix}\mathbf{t}_1 & \mathbf{t}_2 & \cdots & \mathbf{t}_Q\end{bmatrix} \quad \mathbf{P} = \begin{bmatrix}\mathbf{p}_1 & \mathbf{p}_2 & \cdots & \mathbf{p}_Q\end{bmatrix}$$

$$F(\mathbf{W}) = \|\mathbf{T} - \mathbf{W}\mathbf{P}\|^2 = \|\mathbf{E}\|^2$$

$$\|\mathbf{E}\|^2 = \sum_i \sum_j e_{ij}^2$$

Pseudoinverse Rule (3)

Minimize:

$$F(\mathbf{W}) = \|\mathbf{T} - \mathbf{W}\mathbf{P}\|^2 = \|\mathbf{E}\|^2$$

If an inverse exists for $\mathbf{P}$, $F(\mathbf{W})$ can be made zero:

$$\mathbf{W} = \mathbf{T}\mathbf{P}^{-1}$$

However, this is rarely possible.

Dimension of the matrix P = R*Q where
R – no of input components and Q – no of input/target pairs

Usually R and Q are not equal and hence P is not a square matrix and no exact inverse does not exist.

Pseudoinverse Rule (4)

When an inverse does not exist $F(\mathbf{W})$ can be minimized using the pseudoinverse:

$$\boxed{\begin{aligned} \mathbf{W} &= \mathbf{T}\mathbf{P}^{+} \\ \mathbf{P}^{+} &= (\mathbf{P}^{T}\mathbf{P})^{-1}\mathbf{P}^{T} \end{aligned}}$$

$$\Uparrow$$

R (# rows of P) > Q (# columns of P),
Columns of P are independent

Relationship to the Hebb Rule

Hebb Rule

$$\mathbf{W} = \mathbf{T}\mathbf{P}^T$$

Pseudoinverse Rule

$$\mathbf{W} = \mathbf{T}\mathbf{P}^+$$

$$\mathbf{P}^+ = (\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T$$

If the prototype patterns are orthonormal:

$$\mathbf{P}^T\mathbf{P} = \mathbf{I}$$

$$\mathbf{P}^+ = (\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T = \mathbf{P}^T$$

Example (Apple/Banana)

$$\left\{\mathbf{p}_1 = \begin{bmatrix}-1\\1\\-1\end{bmatrix}, \mathbf{t}_1 = \begin{bmatrix}-1\end{bmatrix}\right\} \quad \left\{\mathbf{p}_2 = \begin{bmatrix}1\\1\\-1\end{bmatrix}, \mathbf{t}_2 = \begin{bmatrix}1\end{bmatrix}\right\}$$

$$\mathbf{W} = \mathbf{T}\mathbf{P}^{+} = \begin{bmatrix}-1 & 1\end{bmatrix}\left(\begin{bmatrix}-1 & 1\\1 & 1\\-1 & -1\end{bmatrix}\right)^{+}$$

$$\mathbf{P}^{+} = (\mathbf{P}^{T}\mathbf{P})^{-1}\mathbf{P}^{T} = \begin{bmatrix}3 & 1\\1 & 3\end{bmatrix}^{-1}\begin{bmatrix}-1 & 1 & -1\\1 & 1 & -1\end{bmatrix} = \begin{bmatrix}-0.5 & 0.25 & -0.25\\0.5 & 0.25 & -0.25\end{bmatrix}$$

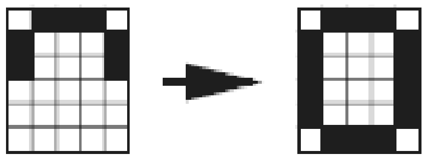
$$\mathbf{W} = \mathbf{T}\mathbf{P}^{+} = \begin{bmatrix}-1 & 1\end{bmatrix}\begin{bmatrix}-0.5 & 0.25 & -0.25\\0.5 & 0.25 & -0.25\end{bmatrix} = \begin{bmatrix}1 & 0 & 0\end{bmatrix}$$

$$\mathbf{W}\mathbf{p}_1 = \begin{bmatrix}1 & 0 & 0\end{bmatrix}\begin{bmatrix}-1\\1\\-1\end{bmatrix} = \begin{bmatrix}-1\end{bmatrix} \qquad \mathbf{W}\mathbf{p}_2 = \begin{bmatrix}1 & 0 & 0\end{bmatrix}\begin{bmatrix}1\\1\\-1\end{bmatrix} = \begin{bmatrix}1\end{bmatrix}$$

Success! outputs and targets are equal

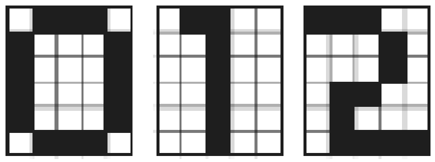
Autoassociative Memory

- Desired output = Input vector (i.e. $\mathbf{t_q} = \mathbf{p_q}$)
- Enables one to retrieve entire memories from only a tiny sample of itself.
- Stores a set of patterns and later recalled them.
- And also corrupted patterns are provided as input.



Corrupted pattern Correct pattern

Example



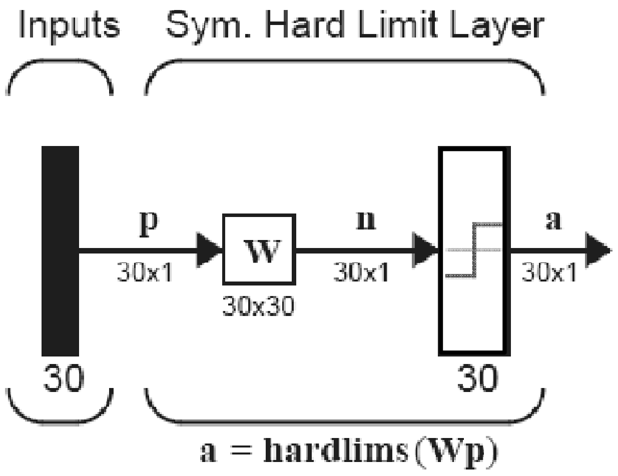
$\mathbf{p}_1, \mathbf{t}_1 \quad \mathbf{p}_2, \mathbf{t}_2 \quad \mathbf{p}_3, \mathbf{t}_3$

Displayed in 6×5 grid

White square will be represented as -1

Black square will be represented as 1

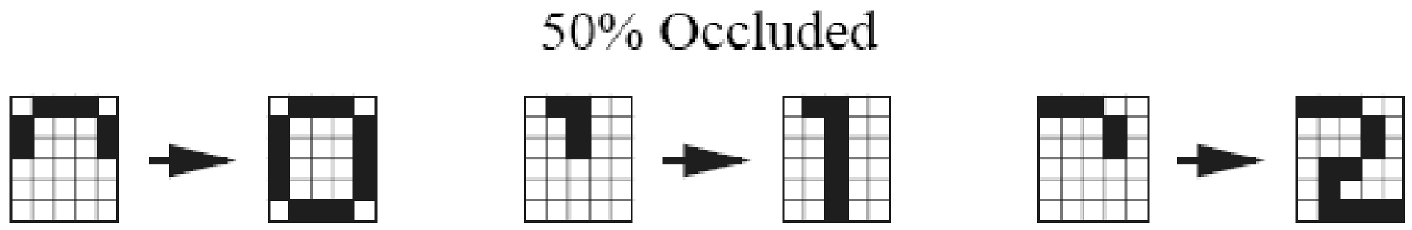
$$\mathbf{p}_1 = \begin{bmatrix} -1 & 1 & 1 & 1 & 1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & -1 & \ldots & 1 & -1 \end{bmatrix}^T$$



$$\mathbf{W} = \mathbf{p}_1\mathbf{p}_1^T + \mathbf{p}_2\mathbf{p}_2^T + \mathbf{p}_3\mathbf{p}_3^T$$

Since there are only two allowable values, symmetrical hard limit transfer function replaces the linear transfer function.

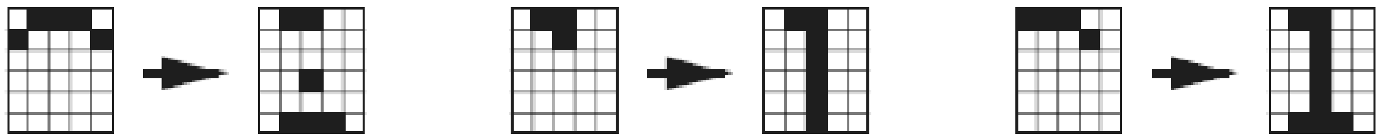
Test 1



- Lower half of the pattern is occluded.
- In each case correct pattern is produced by the network.

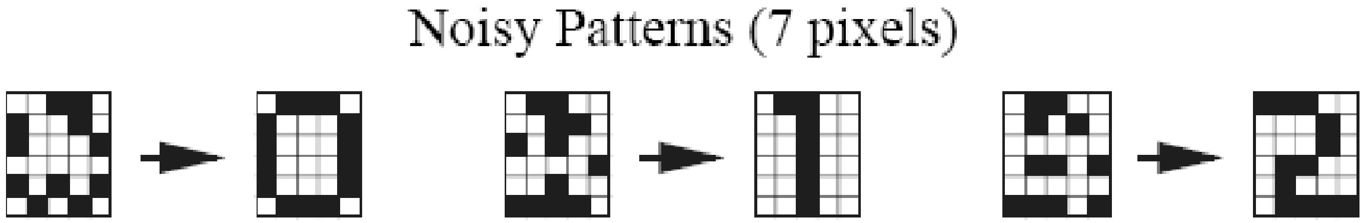
Test 2

67% Occluded



- Removed the lower two-thirds of each pattern.
- Only the digit 1 is recovered.
- Common problem in associative memories.

Test 3



- Presented noisy version of each pattern.
- Changed randomly selected seven elements of each pattern.
- All the patterns were correctly recovered.

Variations of Hebbian Learning (1)

$$\boxed{\text{Basic Rule:} \quad \mathbf{W}^{new} = \mathbf{W}^{old} + \mathbf{t}_q \mathbf{p}_q^T}$$

- Weight matrix may have very large elements if there are many prototype patterns.
- A positive parameter α (<1) called the learning rate, can be used to limit the amount of increase in the weight matrix elements.

$$\boxed{\text{Learning Rate:} \quad \mathbf{W}^{new} = \mathbf{W}^{old} + \alpha \mathbf{t}_q \mathbf{p}_q^T}$$

Variations of Hebbian Learning (2)

$$\boxed{\text{Learning Rate:}\quad \mathbf{W}^{new} = \mathbf{W}^{old} + \alpha \mathbf{t}_q \mathbf{p}_q^T}$$

- We can add a decay term to remember the most recent inputs more clearly.

1 ??????

$$\boxed{\text{Smoothing:}\quad \mathbf{W}^{new} = \mathbf{W}^{old} + \alpha \mathbf{t}_q \mathbf{p}_q^T - \gamma \mathbf{W}^{old} = (1-\overset{\uparrow}{\gamma})\mathbf{W}^{old} + \alpha \mathbf{t}_q \mathbf{p}_q^T}$$

- Remember the most recent inputs more clearly.
- γ - positive constant (<1)
- As γ approaches 1, the learning law quickly forget old inputs and remember only the most recent patterns.

Delta Rule

Delta Rule: $\mathbf{W}^{new} = \mathbf{W}^{old} + \alpha(\mathbf{t}_q - \mathbf{a}_q)\mathbf{p}_q^T$

- Known also as Widrow-Hoff algorithm.
- Adjusts the weights so as to minimize the mean square error.
- Produces the same results as the pseudoinverse rule (which minimizes the sum of squares of errors).
- Advantage – Can update the weights after each new input pattern is presented (whereas the psedoinverse computes the weights at once).
- Adapt to the changing environment.