Associative Learning: Unsupervised Hebbian Learning

Rojas, p. 109

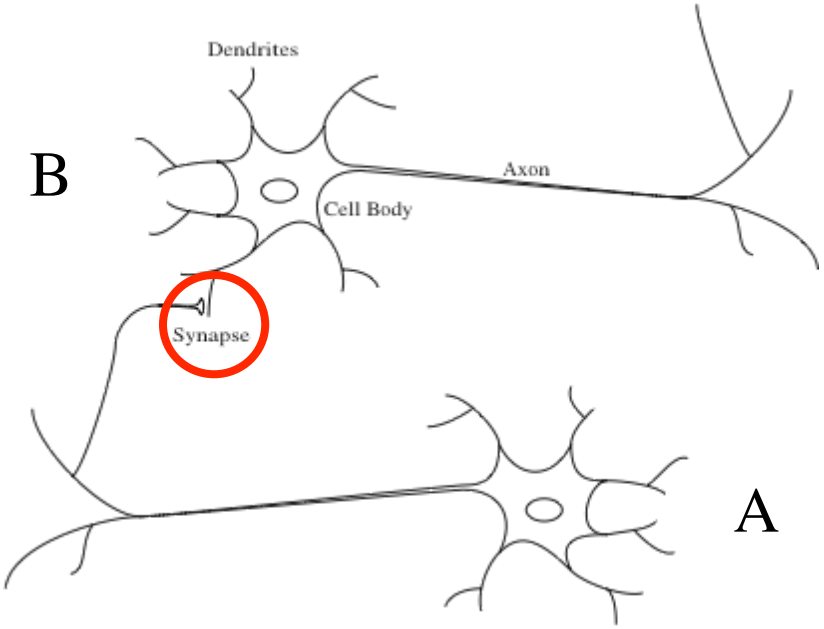
- "We will make a distinction between two classes of unsupervised learning: reinforcement and competitive learning.
  - In the first method each input produces a reinforcement of the network weights in such a way as to enhance the reproduction of the desired output. Hebbian learning is an example of a reinforcement rule that can be applied in this case."

- Note that this is not "reinforcement learning" in the AI sense (e.g. the Temporal Difference method).

Hebb's Postulate

"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."

D. O. Hebb, 1949



Donald Hebb, 1904-1985

More from Hebb, 1949

- "**The general idea is an old one**, that any two cells or systems of cells that are repeatedly active at the same time will tend to become 'associated', so that activity in one facilitates activity in the other." (Hebb 1949, p. 70)

- "When one cell repeatedly assists in firing another, the axon of the first cell develops **synaptic knobs** (or enlarges them if they already exist) in contact with the soma of the second cell." (Hebb 1949, p. 63)

Anticipations

- William James, 1890: “When two brain processes are active together, or in immediate succession, one of them, on reoccurring, tends to propagate its excitement into the other.”

From Rojas, p. 21

- One of the most popular learning algorithms for artificial neural networks is Hebbian learning. The efficiency of synapses is increased any time the two cells which are connected through this synapse fires simultaneously and is decreased when the firing states of the two cells are uncorrelated.

- The **NMDA receptors** act as coincidence detectors of presynaptic and postsynaptic activity, which in turn leads to greater synaptic efficiency.

From Rojas, p. 21

- NMDA receptors are ionic channels permeable for different kinds of molecules, like sodium, calcium, or potassium ions. These channels are blocked by a magnesium ion in such a way that the permeability for sodium and calcium is low.
- If the cell is brought up to a certain excitation level, the ionic channels lose the magnesium ion and become unblocked. The permeability for Ca^{2+} ions increases immediately. Through the flow of calcium ions, a chain of reactions is started which produces a **durable change** of the threshold level of the cell [420, 360].

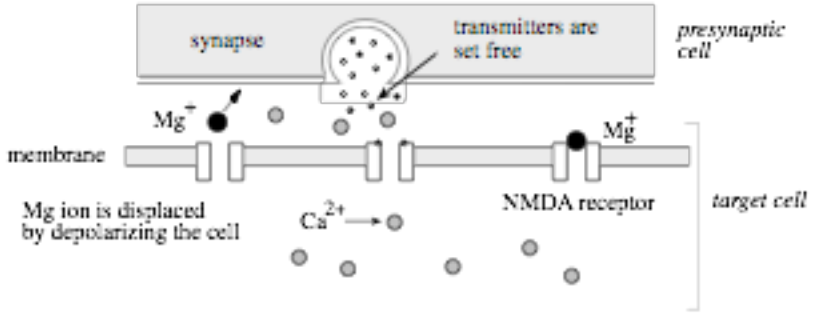


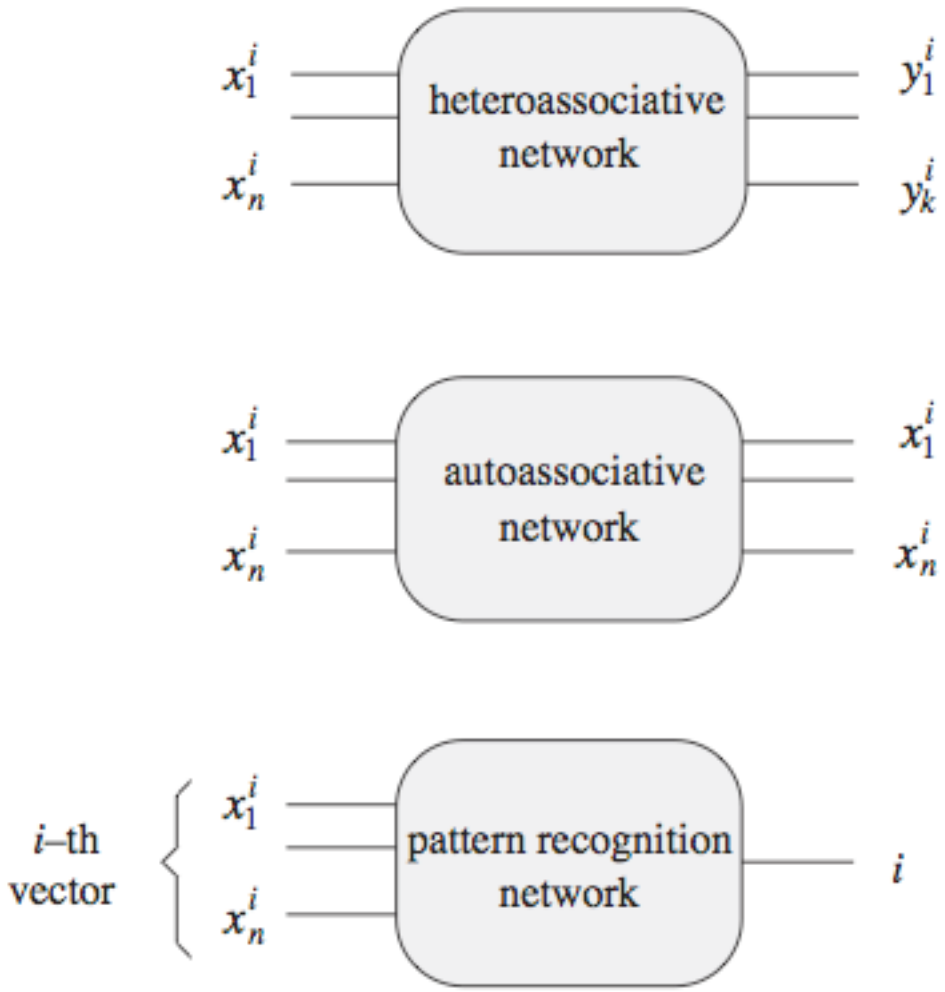
Fig. 1.13. Unblocking of an NMDA receptor

Rojas, p. 312

tinguish between three overlapping kinds of associative networks [294, 255]:

- *Heteroassociative networks* map m input vectors $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^m$ in n-dimensional space to m output vectors $\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^m$ in k-dimensional space, so that $\mathbf{x}^i \mapsto \mathbf{y}^i$. If $\|\tilde{\mathbf{x}} - \mathbf{x}^i\|^2 < \varepsilon$ then $\tilde{\mathbf{x}} \mapsto \mathbf{y}^i$. This should be achieved by the learning algorithm, but becomes very hard when the number m of vectors to be learned is too high.
- *Autoassociative networks* are a special subset of the heteroassociative networks, in which each vector is associated with itself, i.e., $\mathbf{y}^i = \mathbf{x}^i$ for $i = 1, \ldots, m$. The function of such networks is to correct noisy input vectors.
- *Pattern recognition networks* are also a special type of heteroassociative networks. Each vector $\mathbf{x}^i$ is associated with the scalar i. The goal of such a network is to identify the 'name' of the input pattern.

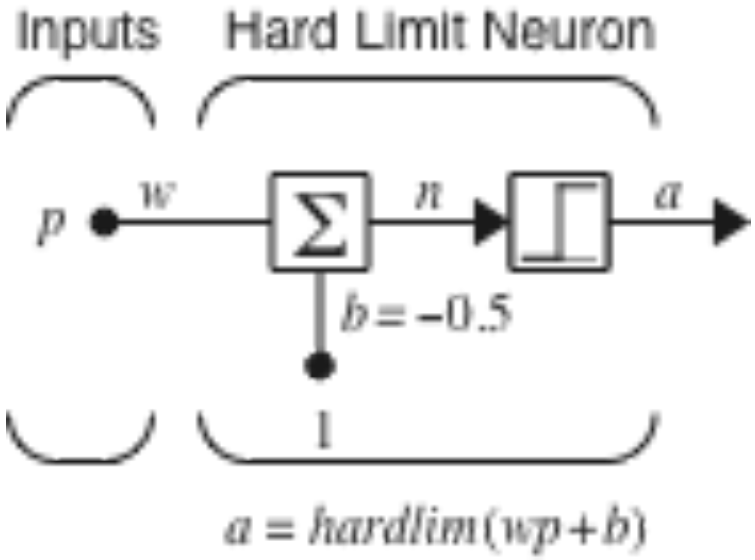
Rojas, p. 313


x_1^i
x_n^i
heteroassociative network
y_1^i
y_k^i
x_1^i
x_n^i
autoassociative network
x_1^i
x_n^i
i–th vector
x_1^i
x_n^i
pattern recognition network
i

Linear Associative Networks

- James Anderson, 1972
- Teuvo Kohonen, 1972

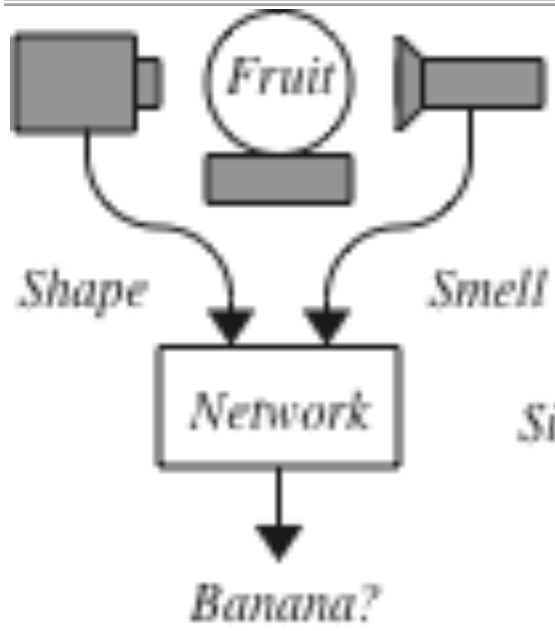
Simple Associative Network

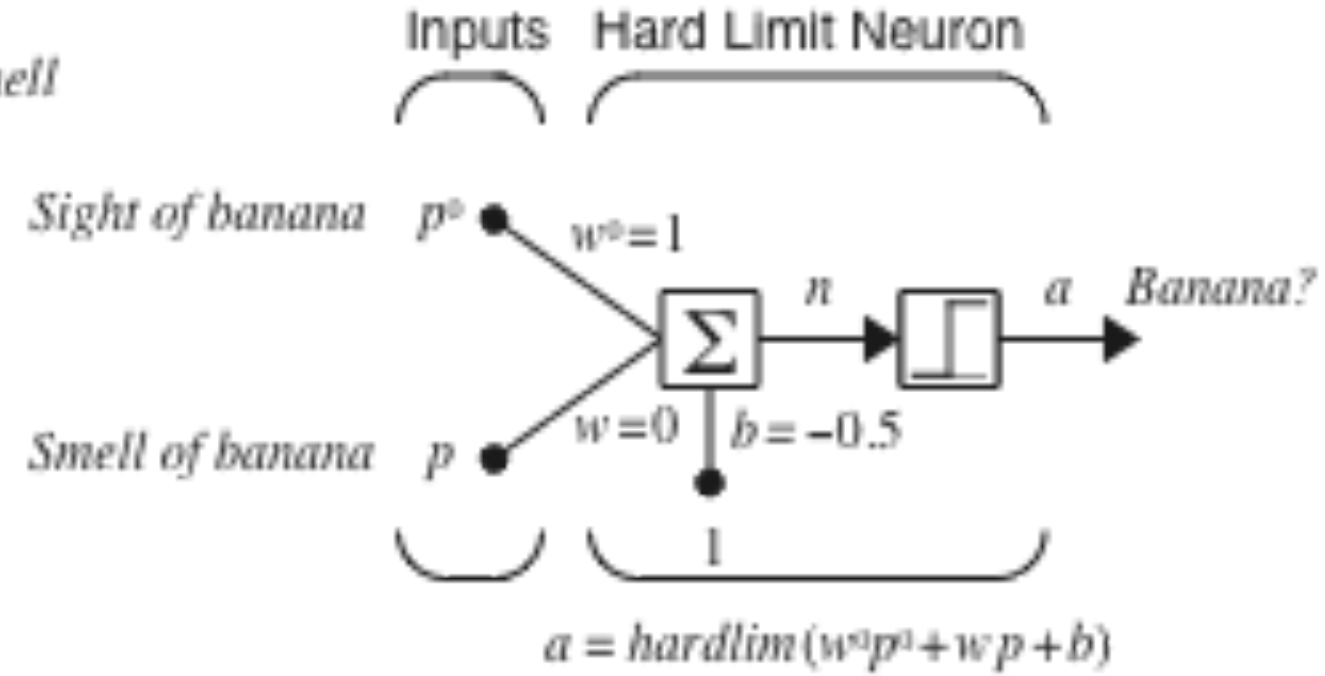


$$a = hardlim(wp + b) = hardlim(wp - 0.5)$$

$$p = \begin{cases} 1, & \text{stimulus} \\ 0, & \text{no stimulus} \end{cases} \qquad a = \begin{cases} 1, & \text{response} \\ 0, & \text{no response} \end{cases}$$

Banana Associator





Unconditioned Stimulus

$$p^0 = \begin{cases} 1, & \text{shape detected} \\ 0, & \text{shape not detected} \end{cases}$$

Conditioned Stimulus

$$p = \begin{cases} 1, & \text{smell detected} \\ 0, & \text{smell not detected} \end{cases}$$

Unsupervised Hebb Rule

$$w_{ij}(\mathrm{q}) = w_{ij}(q-1) + \alpha a_i(q) p_j(q)$$

Vector Form:

$$\mathbf{W}(q) = \mathbf{W}(q-1) + \alpha \mathbf{a}(q)\mathbf{p}^T(q)$$

Note that weights can decrease as well as increase.

Training Sequence:

$$\mathbf{p}(1), \mathbf{p}(2), \ldots, \mathbf{p}(Q)$$

Banana Recognition Example

Initial Weights:

$$w^0 = 1, w(0) = 0$$

Training Sequence:

$$\{p^0(1) = 0, p(1) = 1\}, \{p^0(2) = 1, p(2) = 1\}, \ldots$$

$$\alpha = 1$$

$$w(q) = w(q-1) + a(q)p(q)$$

First Iteration (sight fails):

$$\begin{aligned} a(1) &= hardlim(w^0 p^0(1) + w(0)p(1) - 0.5) \\ &= hardlim(1 \cdot 0 + 0 \cdot 1 - 0.5) = 0 \quad \text{(no response)} \end{aligned}$$

$$w(1) = w(0) + a(1)p(1) = 0 + 0 \cdot 1 = 0$$

Example

Second Iteration (sight works):

$$a(2) = hardlim(w^0 p^0(2) + w(1)p(2) - 0.5)$$
$$= hardlim(1 \cdot 1 + 0 \cdot 1 - 0.5) = 1 \quad \text{(banana)}$$

$$w(2) = w(1) + a(2)p(2) = 0 + 1 \cdot 1 = 1$$

Third Iteration (sight fails):

$$a(3) = hardlim\,(w^0 p^0(3) + w(2)\,p(3) - 0.5)$$
$$= hardlim\,(1 \cdot 0 + 1 \cdot 1 - 0.5) = 1 \quad \text{(banana)}$$

$$w(3) = w(2) + a(3)p(3) = 1 + 1 \cdot 1 = 2$$

Banana will now be detected if either sensor works.

Problems with Hebb Rule

- Weights can become arbitrarily large
- There is no mechanism for weights to decrease

Hebb Rule with Decay

$$\mathbf{W}(q) = \mathbf{W}(q-1) + \alpha \mathbf{a}(q)\mathbf{p}^T(q) - \gamma \mathbf{W}(q-1)$$

$$\boxed{\mathbf{W}(q) = (1-\gamma)\mathbf{W}(q-1) + \alpha \mathbf{a}(q)\mathbf{p}^T(q)}$$

This keeps the weight matrix from growing without bound, which can be demonstrated by setting both a_i and p_j to 1:

$$w_{ij}^{max} = (1-\gamma) w_{ij}^{max} + \alpha a_i p_j$$

$$w_{ij}^{max} = (1-\gamma) w_{ij}^{max} + \alpha$$

$$w_{ij}^{max} = \frac{\alpha}{\gamma}$$

Example: Banana Associator

$\alpha = 1$ $\gamma = 0.1$

First Iteration (sight fails):

$$a(1) = hardlim(w^0 p^0(1) + w(0)p(1) - 0.5)$$
$$= hardlim(1 \cdot 0 + 0 \cdot 1 - 0.5) = 0 \quad \text{(no response)}$$

$$w(1) = w(0) + a(1)p(1) - 0.1w(0) = 0 + 0 \cdot 1 - 0.1(0) = 0$$

Second Iteration (sight works):

$$a(2) = hardlim(w^0 p^0(2) + w(1)p(2) - 0.5)$$
$$= hardlim(1 \cdot 1 + 0 \cdot 1 - 0.5) = 1 \quad \text{(banana)}$$

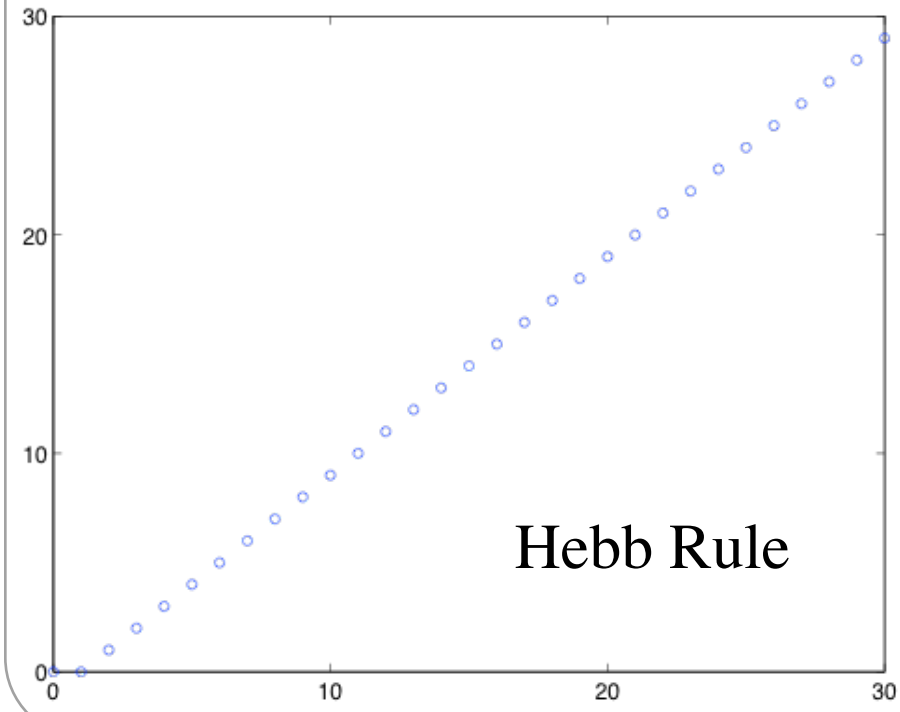
$$w(2) = w(1) + a(2)p(2) - 0.1w(1) = 0 + 1 \cdot 1 - 0.1(0) = 1$$

Example

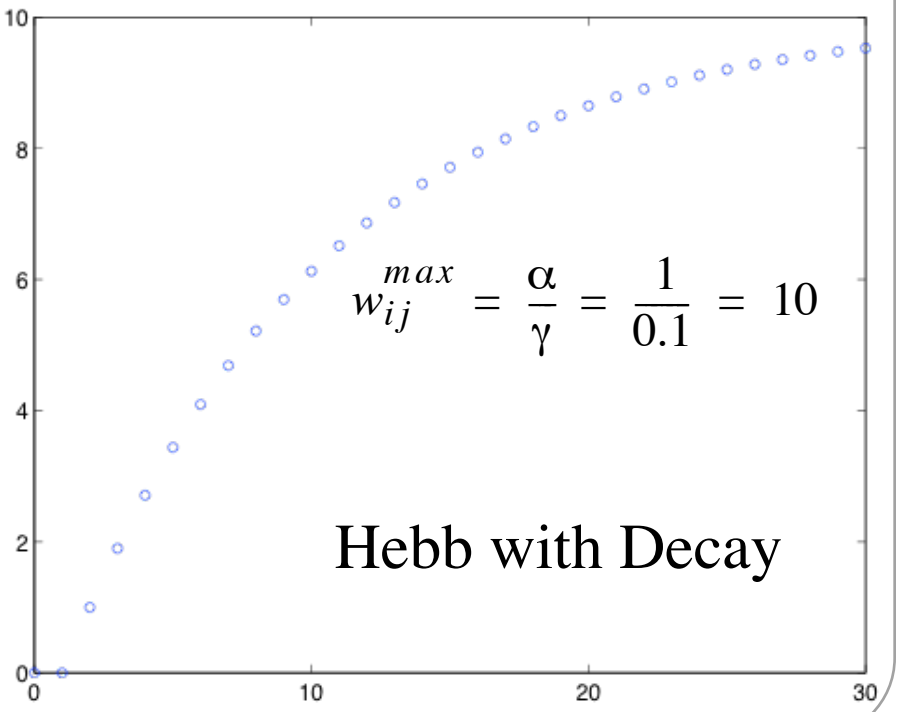
Third Iteration (sight fails):

$$a(3) = hardlim\left(w^0 p^0(3) + w(2)\,p(3) - 0.5\right)$$
$$= hardlim\left(1 \cdot 0 + 1 \cdot 1 - 0.5\right) = 1 \quad \text{(banana)}$$

$$w(3) = w(2) + a(3)p(3) - 0.1w(3) = 1 + 1 \cdot 1 - 0.1(1) = 1.9$$



Hebb Rule



$$w_{ij}^{max} = \frac{\alpha}{\gamma} = \frac{1}{0.1} = 10$$

Hebb with Decay

Problem of Hebb with Decay

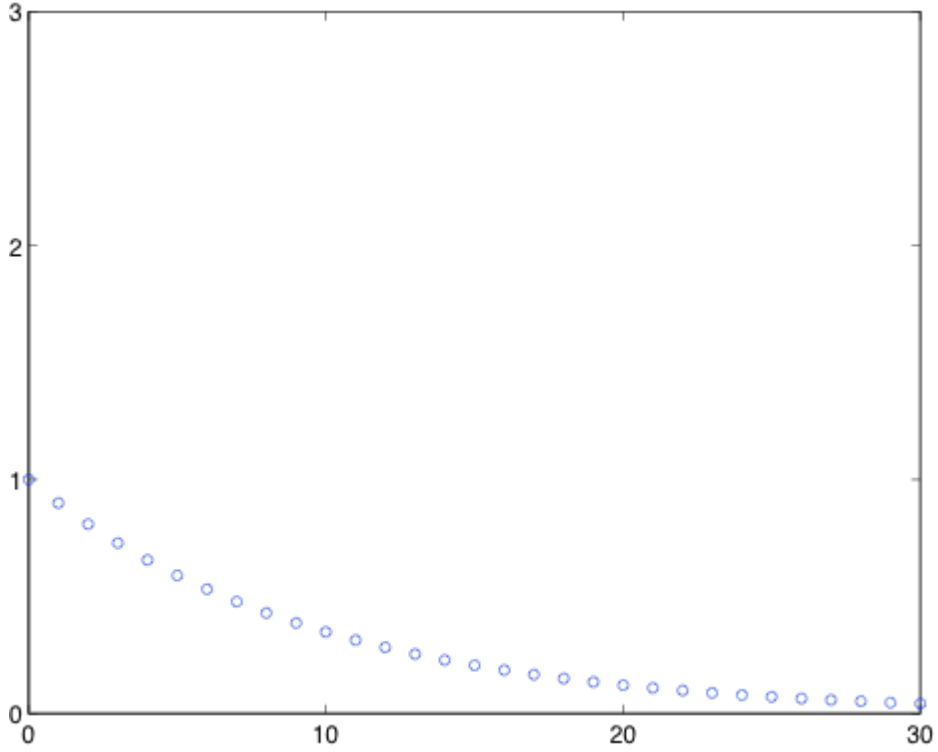
- Associations will decay away if stimuli are not occasionally presented.

If $a_i = 0$, then

$$w_{ij}(q) = (1 - \gamma) w_{ij}(q - 1)$$

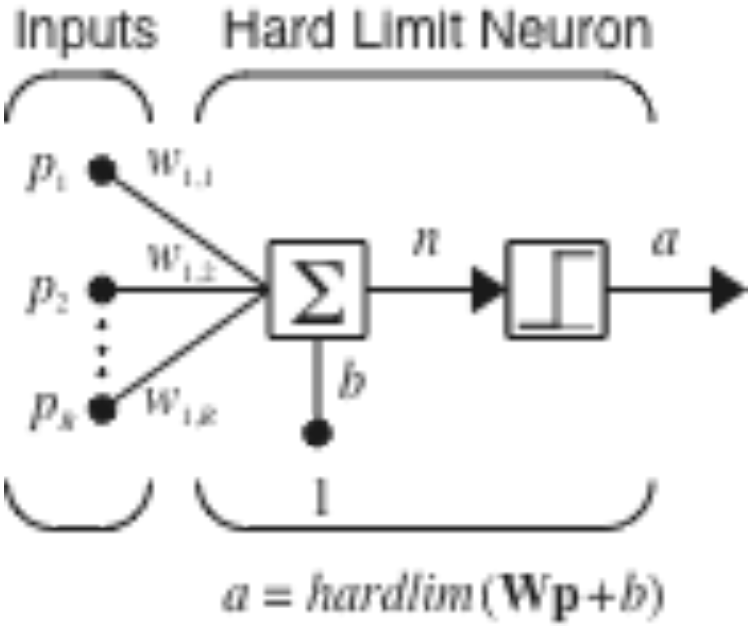
If $\gamma = 0$, this becomes

$$w_{ij}(q) = (0.9) w_{ij}(q - 1)$$



Therefore the weight decays by 10% at each iteration where there is no stimulus.

Instar (Recognition Network)



Instar Operation

$$a = hardlim\,(\mathbf{W}\mathbf{p} + b) = hardlim({}_1\mathbf{w}^T\mathbf{p} + b)$$

The instar will be active when

$${}_1\mathbf{w}^T\mathbf{p} \geq -b$$

or

$${}_1\mathbf{w}^T\mathbf{p} = \|{}_1\mathbf{w}\|\|\mathbf{p}\|\cos\theta \geq -b$$

For normalized vectors, the largest inner product occurs when the angle between the weight vector and the input vector is zero -- the input vector is equal to the weight vector.

The rows of a weight matrix represent patterns to be recognized.

Vector Recognition

If we set

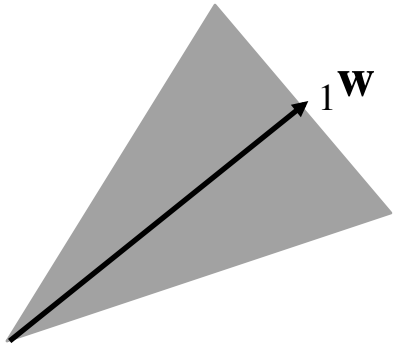
$$b = -\|_1\mathbf{w}\|\|\mathbf{p}\|$$

the instar will only be active when $\theta = 0$.

If we set

$$b > -\|_1\mathbf{w}\|\|\mathbf{p}\|$$

the instar will be active for a range of angles.



As b is increased, the more patterns there will be (over a wider range of θ) which will activate the instar.

Instar Rule

Hebb with Decay

$$w_{ij}(q) = w_{ij}(q-1) + \alpha a_i(q) p_j(q)$$

Modify so that learning and forgetting will only occur when the neuron is active - Instar Rule:

$$w_{ij}(q) = w_{ij}(q-1) + \alpha a_i(q) p_j(q) - \underbrace{\gamma a_i(q) w_{ij}(q-1)}_{\text{“forgetting” term}}$$

or

$$w_{ij}(q) = w_{ij}(q-1) + \alpha a_i(q)(p_j(q) - w_{ij}(q-1))$$

(*if* we make the decay rate γ equal to the learning rate α)

Vector Form:

$$_i\mathbf{w}(q) = {}_i\mathbf{w}(q-1) + \alpha a_i(q)(\mathbf{p}(q) - {}_i\mathbf{w}(q-1))$$

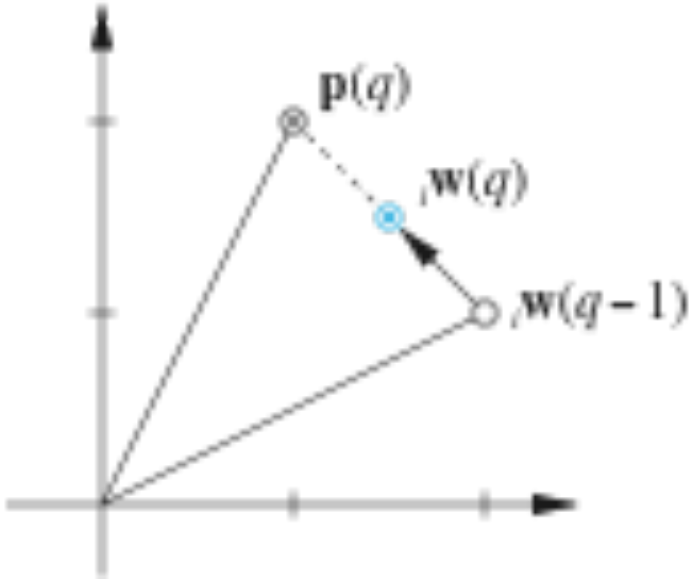
Graphical Representation

For the case where the instar is active ($a_i = 1$):

$$ {}_i\mathbf{w}(q) = {}_i\mathbf{w}(q-1) + \alpha(\mathbf{p}(q) - {}_i\mathbf{w}(q-1)) $$

or

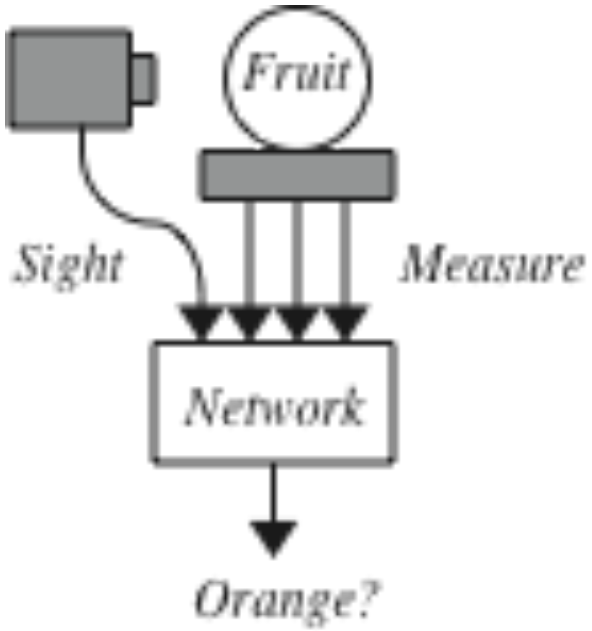
$$ {}_i\mathbf{w}(q) = (1-\alpha){}_i\mathbf{w}(q-1) + \alpha\mathbf{p}(q) $$



For the case where the instar is inactive ($a_i = 0$):

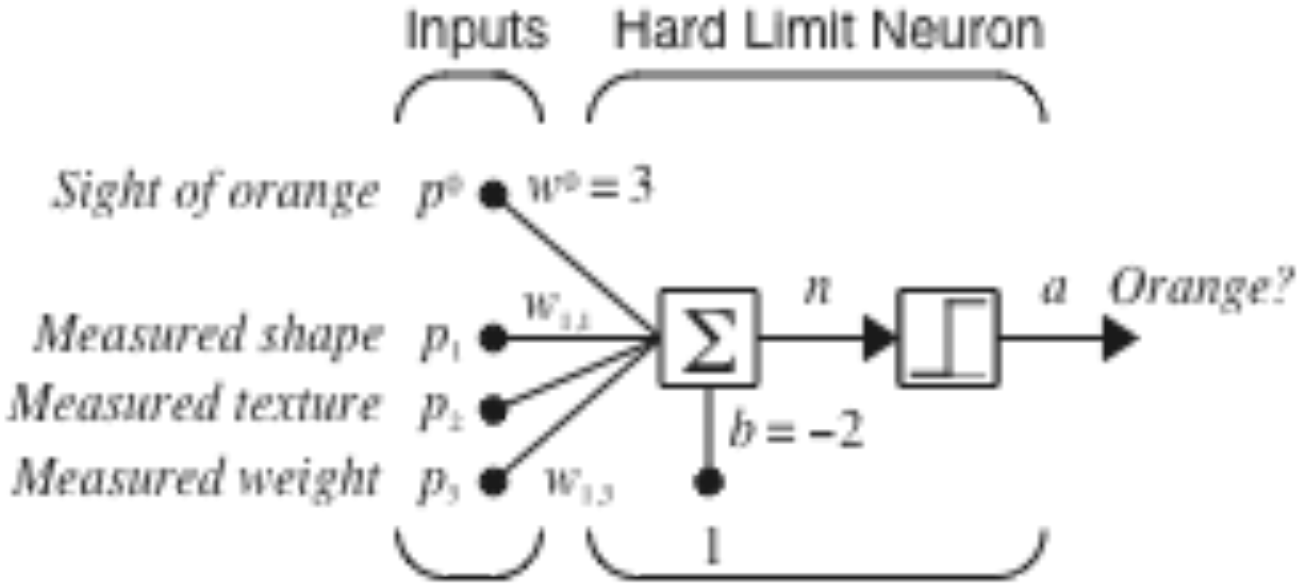
$$ {}_i\mathbf{w}(q) = {}_i\mathbf{w}(q-1) $$

Example



$$p^0 = \begin{cases} 1, & \text{orange detected visually} \\ 0, & \text{orange not detected} \end{cases}$$

$$\mathbf{p} = \begin{bmatrix} shape \\ texture \\ weight \end{bmatrix}$$



$$a = hardlim(w^1 p^1 + \mathbf{W}\mathbf{p} + b)$$

Training

$$\mathbf{W}(0) = {}_1\mathbf{w}^T(0) = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$$

$$\left\{ p^0(1) = 0, \mathbf{p}(1) = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} \right\}, \left\{ p^0(2) = 1, \mathbf{p}(2) = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} \right\}, \ldots$$

First Iteration (α=1):

$$a(1) = hardlim(w^0 p^0(1) + \mathbf{W}\mathbf{p}(1) - 2)$$

$$a(1) = hardlim\left(3 \cdot 0 + \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} - 2\right) = 0 \quad \text{(no response)}$$

$${}_1\mathbf{w}(1) = {}_1\mathbf{w}(0) + a(1)(\mathbf{p}(1) - {}_1\mathbf{w}(0)) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + 0\left(\begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}\right) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Further Training

$$a(2) = hardlim(w^0 p^0(2) + \mathbf{W}\mathbf{p}(2) - 2) = hardlim\left(3 \cdot 1 + \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} - 2\right) = 1 \quad \text{(orange)}$$

$${}_1\mathbf{w}(2) = {}_1\mathbf{w}(1) + a(2)(\mathbf{p}(2) - {}_1\mathbf{w}(1)) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + 1\left(\begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}\right) = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}$$

$$a(3) = hardlim(w^0 p^0(3) + \mathbf{W}\mathbf{p}(3) - 2) = hardlim\left(3 \cdot 0 + \begin{bmatrix} 1 & -1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} - 2\right) = 1 \quad \text{(orange)}$$

$${}_1\mathbf{w}(3) = {}_1\mathbf{w}(2) + a(3)(\mathbf{p}(3) - {}_1\mathbf{w}(2)) = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} + 1\left(\begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} - \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}\right) = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}$$

Orange will now be detected if either set of sensors works.

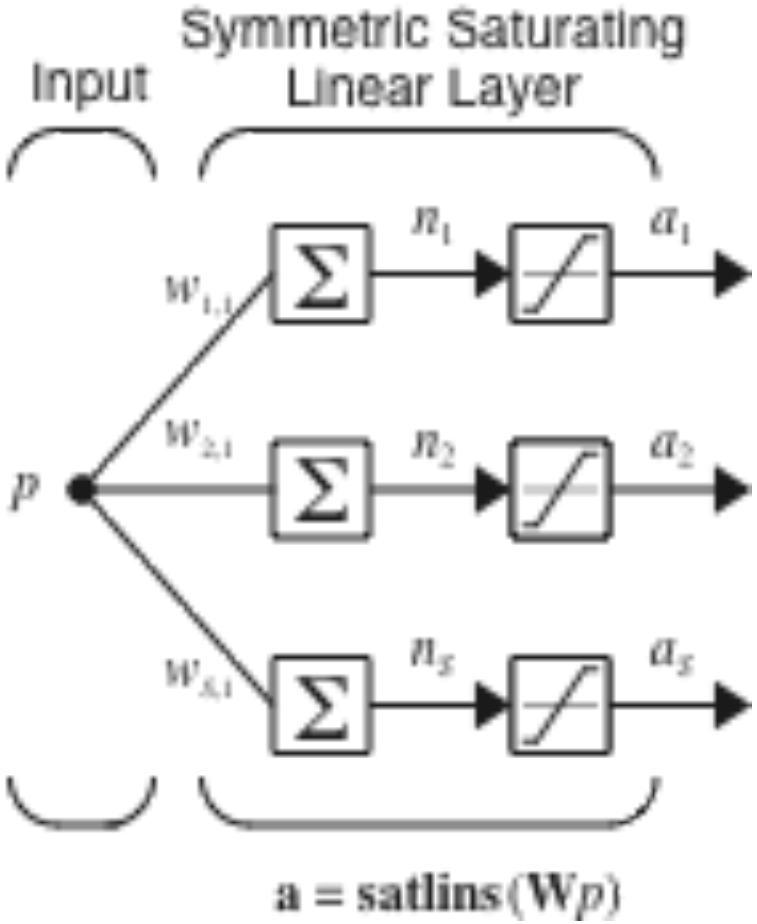
Kohonen Rule

$$ {}_1\mathbf{w}(q) = {}_1\mathbf{w}(q-1) + \alpha(\mathbf{p}(q) - {}_1\mathbf{w}(q-1)), \quad \text{for } i \in X(q) $$

Learning occurs when the neuron's index i is a member of the set $X(q)$. This can be used to train all neurons in a given neighborhood.

(Kohonen = Instar when neighborhood size = 1.)

Outstar (Recall Network)



Outstar Operation

Suppose we want the outstar to recall a certain pattern $\mathbf{a}^*$ whenever the input $p = 1$ is presented to the network. Let

$$\mathbf{W} = \mathbf{a}^*$$

Then, when $p = 1$

$$\mathbf{a} = \mathbf{satlins}(\mathbf{W}p) = \mathbf{satlins}(\mathbf{a}^* \cdot 1) = \mathbf{a}^*$$

and the pattern is correctly recalled.

The columns of a weight matrix represent patterns to be recalled.

Outstar Rule

For the instar rule we made the weight decay term of the Hebb rule proportional to the output of the network. For the outstar rule we make the weight decay term proportional to the input of the network.

$$w_{ij}(q) = w_{ij}(q-1) + \alpha a_i(q)p_j(q) - \gamma p_j(q)w_{ij}(q-1)$$

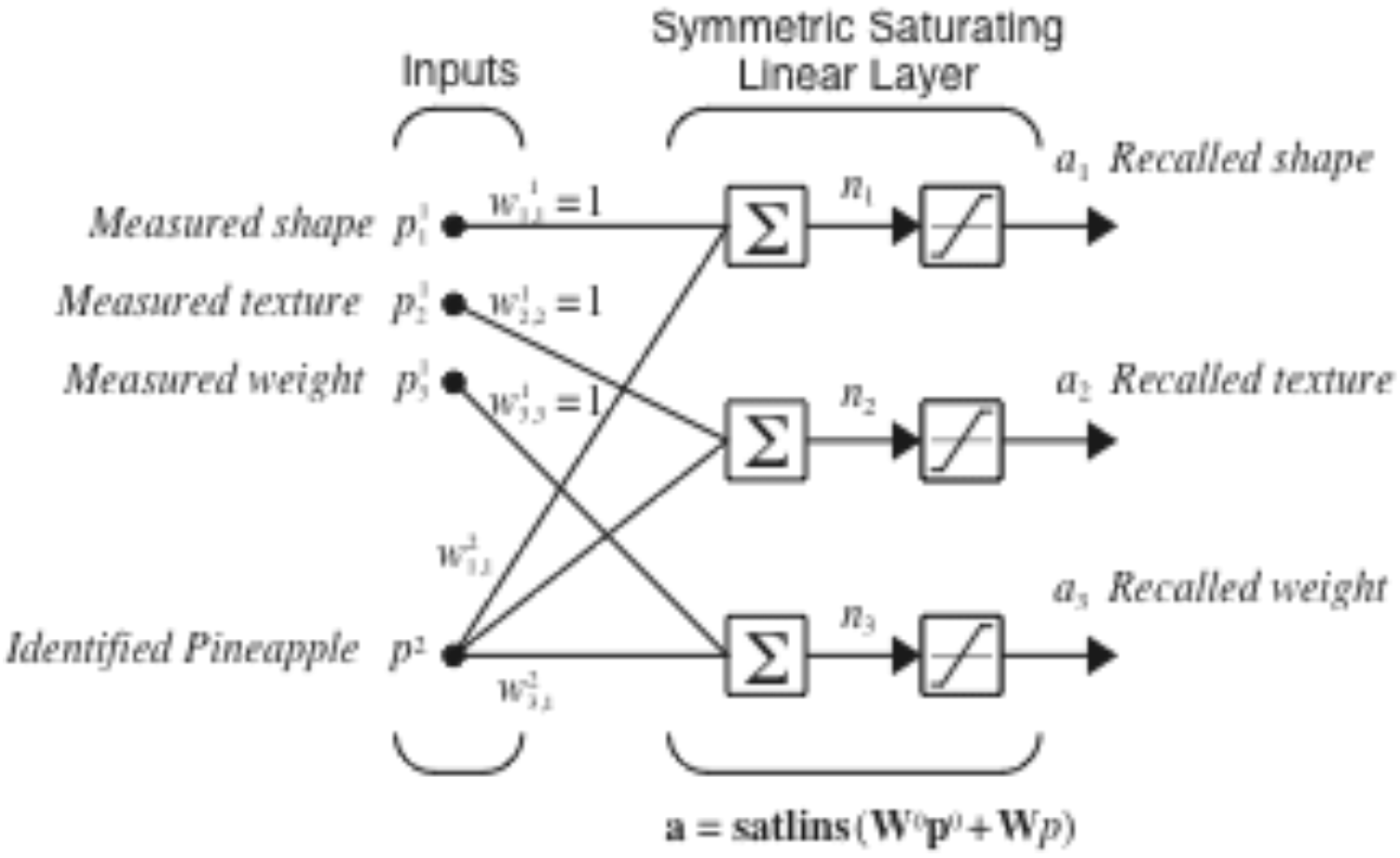
If we make the decay rate γ equal to the learning rate α,

$$w_{ij}(\mathrm{q}) = w_{ij}(q-1) + \alpha(a_i(q) - w_{ij}(q-1))p_j(q)$$

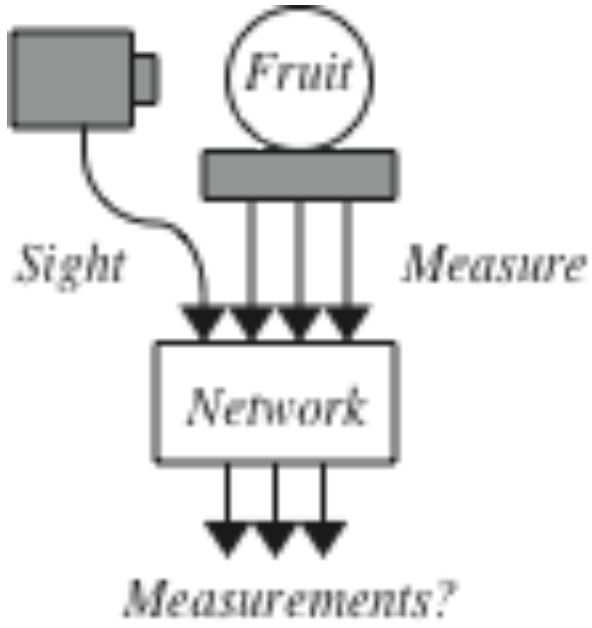
Vector Form:

$$\mathbf{w}_j(q) = \mathbf{w}_j(q-1) + \alpha(\mathbf{a}(q) - \mathbf{w}_j(q-1))p_j(q)$$

Example - Pineapple Recall



Definitions



$$\mathbf{a} = \mathbf{satlins}(\mathbf{W}^0\mathbf{p}^0 + \mathbf{W}p)$$

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

$$\mathbf{p}^0 = \begin{bmatrix} shape \\ texture \\ weight \end{bmatrix} \qquad \mathbf{p}^{pineapple} = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix}$$

$$p = \begin{cases} 1, & \text{if a pineapple can be seen} \\ 0, & \text{otherwise} \end{cases}$$

Iteration 1

$$\left\{ \mathbf{p}^0(1) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, p(1) = 1 \right\}, \left\{ \mathbf{p}^0(2) = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix}, p(2) = 1 \right\}, \ldots$$

$$\alpha = 1$$

$$\mathbf{a}(1) = \mathbf{satlins}\left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} 1 \right) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad \text{(no response)}$$

$$\mathbf{w}_1(1) = \mathbf{w}_1(0) + (\mathbf{a}(1) - \mathbf{w}_1(0))\, p(1) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right) 1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Convergence

$$\mathbf{a}(2) = \mathbf{satlins}\left(\begin{bmatrix}-1\\-1\\1\end{bmatrix} + \begin{bmatrix}0\\0\\0\end{bmatrix}1\right) = \begin{bmatrix}-1\\-1\\1\end{bmatrix} \quad \text{(measurements given)}$$

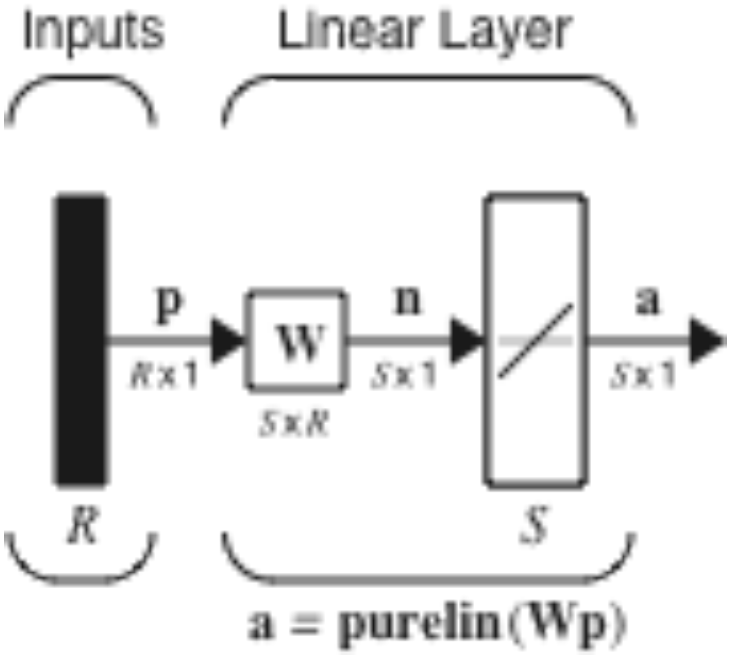
$$\mathbf{w}_1(2) = \mathbf{w}_1(1) + (\mathbf{a}(2) - \mathbf{w}_1(1))p(2) = \begin{bmatrix}0\\0\\0\end{bmatrix} + \left(\begin{bmatrix}-1\\-1\\1\end{bmatrix} - \begin{bmatrix}0\\0\\0\end{bmatrix}\right)1 = \begin{bmatrix}-1\\-1\\1\end{bmatrix}$$

$$\mathbf{a}(3) = \mathbf{satlins}\left(\begin{bmatrix}0\\0\\0\end{bmatrix} + \begin{bmatrix}-1\\-1\\1\end{bmatrix}1\right) = \begin{bmatrix}-1\\-1\\1\end{bmatrix} \quad \text{(measurements recalled)}$$

$$\mathbf{w}_1(3) = \mathbf{w}_1(2) + (\mathbf{a}(2) - \mathbf{w}_1(2))p(2) = \begin{bmatrix}-1\\-1\\1\end{bmatrix} + \left(\begin{bmatrix}-1\\-1\\1\end{bmatrix} - \begin{bmatrix}-1\\-1\\1\end{bmatrix}\right)1 = \begin{bmatrix}-1\\-1\\1\end{bmatrix}$$

Associative Learning: Supervised Hebbian Learning

Linear Associator



$$\mathbf{a} = \mathbf{W}\mathbf{p} \qquad a_i = \sum_{j=1}^{R} w_{ij} p_j$$

Training Set:

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

Supervised Hebb Rule

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

Presynaptic Signal

Postsynaptic Signal

Simplified Form:

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

Supervised Form: Replace output signal with desired target.

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

Matrix Form:

$$\mathbf{W}^{new} = \mathbf{W}^{old} + \mathbf{t}_q \mathbf{p}_q^T$$

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:

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

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

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

(Generally, the targets would be column vectors, and we have *outer* products between the targets and the patterns.)

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 I, input patterns are orthogonal.

$$(\mathbf{p}_q^T\mathbf{p}_k) = 1 \qquad q = k$$

$$= 0 \qquad q \neq k$$

Therefore the network output equals the target:

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

Case II, input patterns are normalized, but not orthogonal.

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

Error term

Example

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

Pseudoinverse Rule - (1)

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

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

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

$$\mathbf{T} = \left[\mathbf{t}_1\ \mathbf{t}_2\ \cdots\ \mathbf{t}_Q\right] \quad \mathbf{P} = \left[\mathbf{p}_1\ \mathbf{p}_2\ \cdots\ \mathbf{p}_Q\right]$$

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

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

Pseudoinverse Rule - (2)

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

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:

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

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

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

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

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

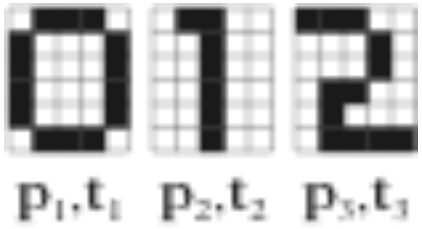
$$\left\{\mathbf{p}_1 = \begin{bmatrix}-1\\1\\-1\end{bmatrix}, \mathbf{t}_1 = \begin{bmatrix}-1\end{bmatrix}\right\} \quad \left\{\mathbf{p}_2 = \begin{bmatrix}1\\1\\-1\end{bmatrix}, \mathbf{t}_2 = \begin{bmatrix}1\end{bmatrix}\right\} \qquad \mathbf{W} = \mathbf{T}\mathbf{P}^+ = \begin{bmatrix}-1 & 1\end{bmatrix}\left(\begin{bmatrix}-1 & 1\\1 & 1\\-1 & -1\end{bmatrix}\right)^+$$

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

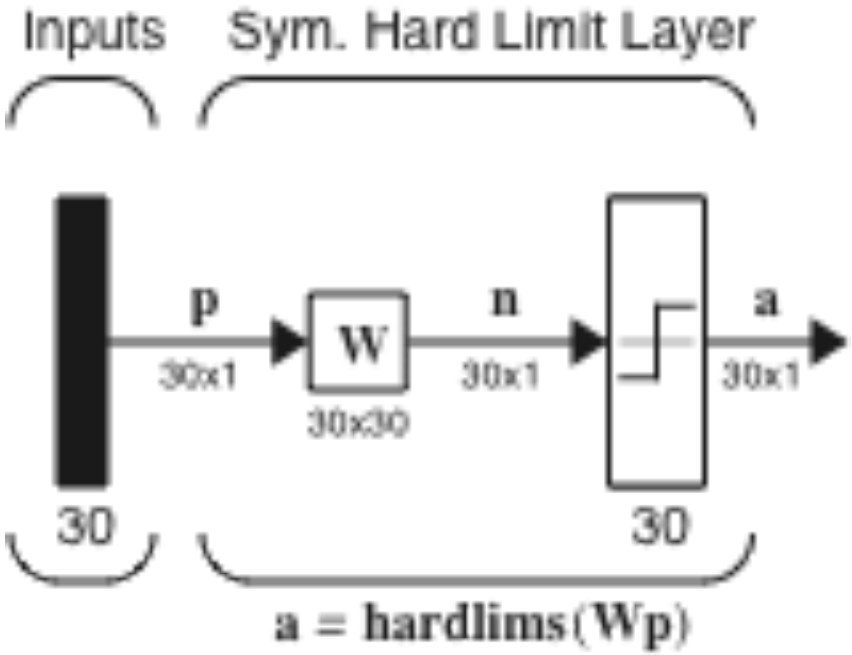
$$\mathbf{W} = \mathbf{T}\mathbf{P}^+ = \begin{bmatrix}-1 & 1\end{bmatrix}\begin{bmatrix}-0.5 & 0.25 & -0.25\\0.5 & 0.25 & -0.25\end{bmatrix} = \begin{bmatrix}1 & 0 & 0\end{bmatrix}$$

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

Autoassociative Memory



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



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

Tests

50% Occluded

67% Occluded

Noisy Patterns (7 pixels)

Variations of Hebbian Learning

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

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

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

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

Unsupervised: $\mathbf{W}^{new} = \mathbf{W}^{old} + \alpha \mathbf{a}_q \mathbf{p}_q^T$

Offshoots

- Hopfield Networks
- PCA Networks
  - Oja's rule, and others