Hebbian Learning
Associative Learning
Please read chapter 4 of Anastasio Book
Synaptic Plasticity

• *Synaptic plasticity* means that weights, i.e. strengths of synaptic connection, can be made to exhibit long-term changes, in a sense, learning or memory.
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
Anticipation of Hebb’s Rule

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.”


William James, 1842-1910
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)
‘Gordon Allport posits additional ideas regarding cell assembly theory and its role in forming engrams, along the lines of the concept of auto-association, described as follows:

"If the inputs to a system cause the same pattern of activity to occur repeatedly, the set of active elements constituting that pattern will become increasingly strongly interassociated. That is, each element will tend to turn on every other element and (with negative weights) to turn off the elements that do not form part of the pattern. To put it another way, the pattern as a whole will become 'auto-associated'. We may call a learned (auto-associated) pattern an engram." (Allport 1985, p. 44)

Hebbian theory has been the primary basis for the conventional view that when analyzed from a holistic level, engrams are neuronal nets or neural networks.’
“Work in the laboratory of Eric Kandel has provided evidence for the involvement of Hebbian learning mechanisms at synapses in the marine gastropod Aplysia californica.

Experiments on Hebbian synapse modification mechanisms at the central nervous system synapses of vertebrates are much more difficult to control than are experiments with the relatively simple peripheral nervous system synapses studied in marine invertebrates. Much of the work on long-lasting synaptic changes between vertebrate neurons (such as long-term potentiation) involves the use of non-physiological experimental stimulation of brain cells. However, some of the physiologically relevant synapse modification mechanisms that have been studied in vertebrate brains do seem to be examples of Hebbian processes. One such study reviews results from experiments that indicate that long-lasting changes in synaptic strengths can be induced by physiologically relevant synaptic activity working through both Hebbian and non-Hebbian mechanisms.”
Long-Term Potentiation (LTP)

A. A weak test pulse (left) evokes the postsynaptic response sketched on the right-hand side of the figure.

B. A strong stimulation sequence (left) triggers postsynaptic firing (right, the peak of the action potential is out of bounds).

C. A test pulse applied some time later evokes a larger postsynaptic response (right; solid line) than the initial response.

Post-Synaptic Strengthening maybe doesn’t *require* Coincidence

http://icwww.epfl.ch/~gerstner//SPNM/node71.html
Cooperativity in the induction of LTP

The previous experiment would also be consistent with a *purely postsynaptic* explanation that claims that the strengthening is solely caused by postsynaptic spike activity. In order to exclude this possibility, a more complicated experiment has to be conducted (Bliss and Collingridge, 1993; Brown et al., 1989).

Synapses at the W channel are strengthened only if both the *presynaptic* site is stimulated via the W electrode and the *postsynaptic* neuron is active due to a *simultaneous* stimulation of the S pathway.

http://icwww.epfl.ch/~gerstner//SPNM/node71.html
Temporal Aspects of LTP
(causality is implicit in Hebb’s postulate?)

Timing requirements between pre- and postsynaptic spikes. Synaptic changes $\Delta w_{ij}$ occur only if presynaptic firing at $t_j(f)$ and postsynaptic activity at $t_i(f)$ occur sufficiently close to each other. Experimentally measured weight changes (circles) as a function of $t_j(f) - t_i(f)$ in milliseconds overlaid on a schematic two-phase learning window (solid line). A positive change (LTP) occurs if the presynaptic spike precedes the postsynaptic one; for a reversed timing, synaptic weights are decreased. Data points redrawn after the experiments of Bi and Poo (1998).

http://icwww.epfl.ch/~gerstner//SPNM/node71.html
“Anti-Hebbian” Possibility

Synapses between parallel fibers and `Purkinje-cells' in the cerebellar-like structure of electric fish, for example, show the **opposite dependence on the relative timing** of presynaptic input and the (so-called `broad') postsynaptic spike.

In this case the synapse is **weakened** if the presynaptic input arrives shortly before the postsynaptic spike (anti-Hebbian plasticity). If the timing is the other way round then the synapse is strengthened. A change in the timing of less than 10 ms can change the effect from depression to potentiation (Bell et al., 1997b).

http://icwww.epfl.ch/~gerstner//SPNM/node71.html
Rojas, p. 21: The NMDA [N-methyl D-aspartate] receptors act as coincidence detectors of presynaptic and postsynaptic activity, which in turn leads to greater synaptic efficiency.

Wikipedia: The NMDA receptor (NMDAR), a glutamate receptor, is the predominant molecular device for controlling synaptic plasticity and memory function.
Durable Change (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.
Three Kinds of Associative Networks
(Rojas, p. 312)

- **Heteroassociative networks** map $m$ input vectors $x_1, x_2, \ldots, x_m$ in $n$-dimensional space to $m$ output vectors $y_1, y_2, \ldots, y_m$ in $k$-dimensional space, so that $x^i \mapsto y^i$. If $\|\tilde{x} - x^i\|^2 < \varepsilon$ then $\tilde{x} \mapsto y^i$. 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., $y^i = x^i$ 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 $x^i$ is associated with the scalar $i$. The goal of such a network is to identify the ‘name’ of the input pattern.
Rojas, p. 313
The many varieties of Hebbian Learning
Simple Associative Network

\[ a = \text{hardlim}(wp + b) = \text{hardlim}(wp - 0.5) \]

\[ p = \begin{cases} 
1, & \text{stimulus} \\
0, & \text{no stimulus} 
\end{cases} \quad a = \begin{cases} 
1, & \text{response} \\
0, & \text{no response} 
\end{cases} \]
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

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

\( a \) designates the output (post-synaptic), \( p \) the input pattern (pre-synaptic)

Vector Form:

\[ W(q) = W(q - 1) + \alpha a(q)p^T(q) \]

Training Sequence:

\( p(1), p(2), \ldots, p(Q) \)
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):

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

\[ w(1) = w(0) + a(1)p(1) = 0 + 0 \cdot 1 = 0 \]
Example

Second Iteration (sight works):

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

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

Third Iteration (sight fails):

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

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

Banana will now be detected if either sensor works.
Problems with Hebb Rule Math

Weights can become arbitrarily large.
Solution: Hebb Rule with Decay $\gamma$

\[ W(q) = W(q-1) + \alpha a(q) p^T(q) - \gamma W(q-1) \]

\[ W(q) = (1 - \gamma)W(q-1) + \alpha a(q) p^T(q) \]

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 with Decay

\[ \alpha = 1 \quad \gamma = 0.1 \]

First Iteration (sight fails):

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

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

Second Iteration (sight works):

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

\[ w(2) = w(1) + a(2)p(2) - 0.1w(1) = 0 + 1 \cdot 1 - 0.1(0) = 1 \]
Example

Third Iteration (sight fails):

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

\[ w(3) = w(2) + a(3) \ p(3) - 0.1 \ w(3) = 1 + 1 \cdot 1 - 0.1(1) = 1.9 \]
Problem of Hebb with Decay

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

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

If \( a_i = 0 \), then

\[ w_{ij}(q) = (0.9)w_{ij}(q - 1) \]

If \( \gamma = 0 \), this becomes

Therefore the weight decays by 10% at each iteration where there is no stimulus.
“Instar” (Recognition Network), Grossberg
Purpose is to learn a single pattern

Essentially like a perceptron, learning to distinguish one central thing from all others.
Instar Operation

\[ a = \text{hardlim}(Wp + b) = \text{hardlim}(w^T p + b) \]

The instar will be active when

\[ w^T p \geq -b \]

or

\[ w^T p = \|w\|\|p\|\cos \theta \geq -b \]

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 by Instar

If we set
\[
b = -\|w\|p\|
\]
the instar will only be active when \(\theta = 0\).

If we set
\[
b > -\|w\|p\|
\]
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 \(\theta\)) 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:

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

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 \( \gamma \) equal to the learning rate \( \alpha \))

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 ($a_i = 1$):

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

or

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

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

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

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

\[ p = \begin{bmatrix} \text{shape} \\
\text{texture} \\
\text{weight} \end{bmatrix} \]

\[ a = \text{hardlim}(w^0 p^0 + \mathbf{W} p + b) \]
Training Instar

\[ \mathbf{W}(0) = \mathbf{1} \mathbf{w}^T(0) = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \]

\[
\begin{cases}
  p^0(1) = 0, \mathbf{p}(1) = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}, \\
  p^0(2) = 1, \mathbf{p}(2) = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}
\end{cases}
\]

First Iteration (\(\alpha=1\)):

\[ a(1) = hardlim(\mathbf{w}^0 p^0(1) + \mathbf{Wp}(1) - 2) \]

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

\[ \mathbf{1w}(1) = \mathbf{1w}(0) + a(1)(\mathbf{p}(1) - \mathbf{1w}(0)) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + 0 \left( \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \]
Further Training of Instar

\[ a(2) = \text{hardlim}(w^0 p^0(2) + Wp(2) - 2) = \text{hardlim}\left(3 \cdot 1 + \begin{bmatrix} 0 & 0 \\ -1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} - 2\right) = 1 \text{ (orange)} \]

\[ _1w(2) = _1w(1) + a(2)(p(2) - _1w(1)) = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + 1 \begin{bmatrix} 1 \\ -1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \]

\[ a(3) = \text{hardlim}(w^0 p^0(3) + Wp(3) - 2) = \text{hardlim}\left(3 \cdot 0 + \begin{bmatrix} 1 & -1 & -1 \\ -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} - 2\right) = 1 \text{ (orange)} \]

\[ _1w(3) = _1w(2) + a(3)(p(3) - _1w(2)) = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} + 1 \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} - \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} \]

Orange will now be detected if \textit{either} set of sensors works.
Kohonen Rule
(more on this topic later)

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

Learning occurs when the neuron’s index \( i \) is a member of the “neighborhood” set \( X(q) \).

**All neurons in the immediate neighborhood learn.**

(Kohonen = Instar when neighborhood size = 1.)
Outstar (Recall Network) Grossberg
The “dual” of an Instar

\[ a = \text{satlins}(W_p) \]
Outstar Operation

Suppose we want the outstar to recall a certain pattern \( a^* \) whenever the input \( p = 1 \) is presented to the network. Let

\[
W = a^*
\]

Then, when \( p = 1 \)

\[
a = \text{satlin}(Wp) = \text{satlin}(a^* \cdot 1) = a^*
\]

and the pattern is correctly recalled.

The columns of a weight matrix represent patterns to be recalled.
Outstar Rule with Weight Decay

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 \( \gamma \) equal to the learning rate \( \alpha \),

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

Vector Form:

\[ w_j(q) = w_j(q-1) + \alpha (a(q) - w_j(q-1))p_j(q) \]
Example - Pineapple Recall

\[
a = \text{satlins}(W^0 p^0 + W p)
\]
Definitions

\[ a = \text{satl ins}(W^0 p^0 + W p) \]

\[ W^0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \]

\[ p^0 = \begin{bmatrix} \text{shape} \\ \text{texture} \\ \text{weight} \end{bmatrix} \quad p^{\text{pineapple}} = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} \]

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

\[
\begin{align*}
\{ p^0(1) &= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, p(1) = 1 \}, \\
p^0(2) &= \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix}, p(2) = 1 \} \quad \ldots
\end{align*}
\]

\[
\alpha = 1
\]

\[
a(1) = \text{satlns} \left( \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad \text{(no response)}
\]

\[
w_1(1) = w_1(0) + (a(1) - w_1(0)) p(1) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}
\]
Convergence

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

\[ w_1(2) = w_1(1) + (a(2) - w_1(1))p(2) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \left( \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right) 1 = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} \]

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

\[ w_1(3) = w_1(2) + (a(2) - w_1(2))p(2) = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} + \left( \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} - \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} \right) 1 = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} \]
Supervised Hebbian Learning
Linear Associator

\[ a = Wp \]

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

Training Set:

\[ \{p_1, t_1\}, \{p_2, t_2\}, \ldots, \{p_Q, t_Q\} \]

\( a \) designates the output,
\( p \) the input pattern
\( t \) the target
Linear Associative Networks

• James Anderson, 1972

• Teuvo Kohonen, 1972
Unsupervised to Supervised Hebb Rule

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

Learning rate \( \alpha \) \[
\text{Presynaptic Signal}
\]

Postsynaptic Signal

Simplified Form:
\[
w_{ij}^{new} = w_{ij}^{old} + \alpha a_{iq} p_{jq}
\]

Supervised Form: Replace output signal with desired target.
\[
w_{ij}^{new} = w_{ij}^{old} + t_{iq} p_{jq}
\]

\( t \) designates the target, \( p \) the input pattern

Matrix Form:
\[
W^{new} = W^{old} + t_q p_q^T
\]
Batch Operation Hebbian Weights

\[ W = t_1 p_1^T + t_2 p_2^T + \cdots + t_Q p_Q^T = \sum_{q=1}^{Q} t_q p_q^T \]  
(Zero Initial Weights)

Matrix Form:

\[
W = \begin{bmatrix}
  t_1 & t_2 & \cdots & t_Q \\
  p_1^T \\
  p_2^T \\
  \vdots \\
  p_Q^T
\end{bmatrix} = TP^T
\]

\[ P = \begin{bmatrix} p_1 & p_2 & \cdots & p_Q \end{bmatrix} \]

\[ T = \begin{bmatrix} t_1 & t_2 & \cdots & t_Q \end{bmatrix} \]

(Generally, targets and patterns are column vectors, thus we have outer products between the targets and the patterns.)
Example

<table>
<thead>
<tr>
<th>Banana</th>
<th>Apple</th>
</tr>
</thead>
</table>

\[
P_1 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}, \quad P_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}
\]

Normalized Prototype Patterns:

\[
P_1 = \begin{bmatrix} -0.5774 \\ 0.5774 \\ -0.5774 \end{bmatrix}, \quad t_1 = [-1]
\]

\[
P_2 = \begin{bmatrix} 0.5774 \\ 0.5774 \\ -0.5774 \end{bmatrix}, \quad t_2 = [1]
\]

Weight Matrix (Hebb Rule):

\[
W = TP^T = \begin{bmatrix} -1 & 1 \end{bmatrix} \begin{bmatrix} -0.5774 & 0.5774 & -0.5774 \\ 0.5774 & 0.5774 & -0.5774 \end{bmatrix} = \begin{bmatrix} 1.1548 & 0 & 0 \end{bmatrix}
\]

Tests:

Banana

\[
Wp_1 = \begin{bmatrix} 1.1548 & 0 & 0 \end{bmatrix} \begin{bmatrix} -0.5774 \\ 0.5774 \\ -0.5774 \end{bmatrix} = \begin{bmatrix} -0.6668 \end{bmatrix}
\]

Apple

\[
Wp_2 = \begin{bmatrix} 0 & 1.1548 & 0 \end{bmatrix} \begin{bmatrix} 0.5774 \\ 0.5774 \\ -0.5774 \end{bmatrix} = \begin{bmatrix} 0.6668 \end{bmatrix}
\]
Performance Analysis

\[ a = W p_k = \left( \sum_{q=1}^{Q} t_q p_q^T p_k \right) = \sum_{q=1}^{Q} t_q (p_q^T p_k) \]

Case I, input patterns are **normalized and orthogonal**.

\[
(p_q^T p_k) = 1 \quad q = k \\
= 0 \quad q \neq k
\]

Therefore the network output equals the target:

\[ a = W p_k = t_k \]

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

\[ a = W p_k = t_k + \sum_{q \neq k} t_q (p_q^T p_k) \]

Error term
Improving Performance
Pseudoinverse Rule - (1)

Performance Index: \( \mathbf{wp}_q = \mathbf{t}_q \quad q = 1, 2, \ldots, Q \)

\[
F(\mathbf{W}) = \sum_{q=1}^{Q} \| \mathbf{t}_q - \mathbf{wp}_q \|^2
\]

Matrix Form:
\[
\mathbf{WP} = \mathbf{T}
\]

\[
\mathbf{T} = \begin{bmatrix} t_1 & t_2 & \cdots & t_Q \end{bmatrix} \quad \mathbf{P} = \begin{bmatrix} p_1 & p_2 & \cdots & p_Q \end{bmatrix}
\]

\[
F(\mathbf{W}) = \| \mathbf{T} - \mathbf{WP} \|^2 = \| \mathbf{E} \|^2
\]

\[
\| \mathbf{E} \|^2 = \sum_{i} \sum_{j} e_{ij}^2
\]
Pseudoinverse Rule - (2)

\[ WP = T \]

Minimize:

\[ F(W) = \| T - WP \|^2 = \| E \|^2 \]

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

\[ W = TP^{-1} \]

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

\[ W = TP^+ \]

\[ P^+ = (P^T P)^{-1} P^T \]
Relationship to the Hebb Rule

Hebb Rule
\[ W = TP^T \]

Pseudoinverse Rule
\[ W = TP^+ \]
\[ P^+ = (P^T P)^{-1} P^T \]

If the prototype patterns are *orthonormal* the two rules are equivalent:
\[ P^T P = I \]
\[ P^+ = (P^T P)^{-1} P^T = P^T \]
Example using Pseudoinverse

\[
\begin{align*}
\{ & \mathbf{p}_1 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}, \mathbf{t}_1 = \begin{bmatrix} -1 \end{bmatrix} \} \\
\{ & \mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}, \mathbf{t}_2 = \begin{bmatrix} 1 \end{bmatrix} \}
\end{align*}
\]

\[
\mathbf{W} = \mathbf{T}\mathbf{P}^+ = \begin{bmatrix} -1 & 1 \end{bmatrix} \begin{bmatrix} -1 & 1 \\ 1 & 1 \\ -1 & -1 \end{bmatrix}^+
\]

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

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

\[
\mathbf{W}\mathbf{p}_1 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} = \begin{bmatrix} -1 \end{bmatrix} \quad \quad \mathbf{W}\mathbf{p}_2 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \end{bmatrix}
\]
Autoassociative Memory Demo

\[ p_1 = \begin{bmatrix} -1 & 1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & 1 & \cdots & 1 & -1 \end{bmatrix}^T \]

\[ W = p_1 p_1^T + p_2 p_2^T + p_3 p_3^T \]  

(Hebb Rule)
Tests

50% Occluded Patterns

67% Occluded Patterns

Noisy Patterns (7 pixels)
Variations of Hebbian Learning

Basic Rule: \( W^{new} = W^{old} + t_q p_q^T \)

Learning Rate: \( W^{new} = W^{old} + \alpha t_q p_q^T \)

Smoothing: \( W^{new} = W^{old} + \alpha t_q p_q^T - \gamma W^{old} = (1 - \gamma)W^{old} + \alpha t_q p_q^T \)

Delta Rule: \( W^{new} = W^{old} + \alpha (t_q - a_q) p_q^T \)

Unsupervised: \( W^{new} = W^{old} + \alpha a_q p_q^T \)
Related Topics to be Studied

• Hopfield Networks

• PCA Networks
  – Oja’s rule, and others