Associative Learning
(Unsupervised Hebbian and others)

Simple Associative Network

Unsupervised Hebb Rule

Banana Associator

Unconditioned Stimulus

Conditioned Stimulus

Unconditioned Stimulus

Conditioned Stimulus

Banana Recognition Example

Initial Weights:

Training Sequence:

First Iteration (sight fails):

Second Iteration (sight works):

Third Iteration (sight fails):

Example

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

---

**Example: Banana Associator**

\[
\begin{align*}
\begin{array}{c|c|c}
\text{stimulus} & \text{response} & \text{weight} \\
\text{banana} & \text{no} & 0.1 \\
\text{banana} & \text{yes} & 1 \\
\end{array}
\end{align*}
\]

First Iteration (sight fails):

\[
a(1) = \text{hardlim}(w_{11}^{0} + w_{12}^{0}) = 0.1 \quad \text{(no response)}
\]

\[
w(1) = w(0) + a(1)p(1) - 0.5w(0) = 0 + 0.1 - 0.5(0) = 0
\]

Second Iteration (sight works):

\[
a(2) = \text{hardlim}(w_{11}^{1} + w_{12}^{1}) = 1 \quad \text{(banana)}
\]

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

---

**Problem of Hebb with Decay**

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

If \( a = 0 \), then

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

If \( || = 0 \), 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.

---

**Hebb Rule with Decay**

\[
\begin{align*}
W(q) &= W(q-1) + (a_i \cdot p_i) (q) - (W(q-1)) \\
W(q) &= (1-W(q-1)) + (a_i \cdot p_i)(q)
\end{align*}
\]

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

\[
\begin{align*}
W_{ij}^\text{max} &= (1-||W_{ij}^\text{max}|| + ||a_i \cdot p_i||) \\
W_{ij}^\text{max} &= (1-||W_{ij}^\text{max}|| + ||a_i \cdot p_i||) \\
W_{ij}^\text{max} &= \frac{1}{2}
\end{align*}
\]

**Example**

Third Iteration (sight fails):

\[
a(3) = \text{hardlim}(w_{13}^{2} + w_{23}^{2} - 0.5) = \text{hardlim}(1 - 0.5) = 0 \quad \text{(banana)}
\]

\[
w(3) = w(2) + a(3)p(3) - 0.1w(2) = 1 + 1 - 0.1(1) = 1.9
\]

- **Hebb Rule**
- **Hebb with Decay**

---

**Instar (Recognition Network)**

\[
a = \text{hardlim}(\sum_{i} W_{ii} \cdot p_i)
\]
Instar Operation

\[ w = \text{hardlim}(Wp + b) = \text{hardlim}(W'p + b) \]

The instar will be active when

\[ w'p = ba \]

or

\[ w'p = \|w'\| \|p\| = ab \]

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

\[ b = -1|\mathbf{w}| \]

the instar will only be active when \( |\mathbf{p}| = 0 \).

If we set

\[ b > -1|\mathbf{w}| \]

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 \( |\mathbf{p}| \) which will activate the instar.

Instar Rule

Hebb with Decay

\[ w_i(q) = w_i(q-1) + \|\mathbf{q}\| \|\mathbf{q}_i\| \]

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

\[ w_i(q) = w_i(q-1) + \|\mathbf{q}\| \|\mathbf{q}_i\| - w_i(q-1) \]

or

\[ w_i(q) = w_i(q-1) + \|\mathbf{q}\| \|\mathbf{q}_i\| - w_i(q-1) \]

Vector Form:

\[ \mathbf{w}(q) = \mathbf{w}(q-1) + \|\mathbf{q}\| \|\mathbf{q}_i\| - \mathbf{w}(q-1) \]

Graphical Representation

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

\[ \mathbf{w}(q) = \mathbf{w}(q-1) + \|\mathbf{q}\| \|\mathbf{q}_i\| - \mathbf{w}(q-1) \]

\[ \mathbf{w}(q) = (1 - \|\mathbf{q}\|) \mathbf{w}(q-1) \]

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

\[ \mathbf{w}(q) = \mathbf{w}(q-1) \]

Example

Training

\[ W(0) = \mathbf{w}(0) = 0 \]

\[ \hat{\mathbf{p}}(1) = 0, \mathbf{p}(1) = \begin{pmatrix} 1 \end{pmatrix}, \mathbf{p}(2) = \begin{pmatrix} 0 \end{pmatrix} \]

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

\[ a(1) = \text{hardlim}(\mathbf{w}(p(1)) + \mathbf{W}(p(1) - 2)) \]

\[ a(1) = \text{hardlim}(0 + 0) = 0 \]

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

\[ a(1) = \text{hardlim}(\mathbf{w}(1) + \mathbf{W}(p(1) - 2)) \]
Further Training

\[ a(2) = \text{hardlim}(a(1)p(2) + Wp(2) - 2) = \text{hardlim}(3 + 0.1 - 2) = 1 \] (orange)

\[ a(2) = \text{hardlim}(a(1)p(2) + Wp(2) - 2) = \text{hardlim}(3 + 0.1 - 2) = 1 \] (orange)

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

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 \( \mathbf{W} = \mathbf{W}^* \).

Then, when \( p = 1 \)

\[ \mathbf{a} = \text{satlim}(\mathbf{W}p) = \text{satlim}(\mathbf{a}^*) = \mathbf{a}^* \]

and the pattern is correctly recalled.

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

Example - Pineapple Recall

Outstar Rule

For the instar rule we make 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.

\[ w_i(j) = w_i(j - 1) + [\text{sat}(p_i)p_i] - [\text{sat}(p_i)p_i]w_i(j - 1) \]

If we make the decay rate equal to the learning rate \( \lambda \),

\[ w_i(j) = w_i(j - 1) + \lambda [\text{sat}(p_i)p_i - w_i(j - 1)] \]

Vector Form:

\[ w_i(j) = w_i(j - 1) + \lambda [\text{sat}(p_i)p_i - w_i(j - 1)] \]

Definitions

\[ a = \text{satlim}(Wp^T + Wp) \]

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

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

\[ p = \begin{bmatrix} 1 \end{bmatrix}, \text{if a pineapple can be seen} \]

\[ 0, \text{otherwise} \]
Iteration 1

\[ p(1) = p(1) = 1 \]
\[ p(2) = p(2) = 1 \]
\[ p = 1 \]

\[ w_1(1) = w_1(0) + (a(1) - w_1(0))p(1) = 1 + 1 = 2 \]

Convergence

\[ a(2) = \text{satlin} \]
\[ w_1(2) = w_1(1) + (a(2) - w_1(1))p(2) = 2 + 1 = 3 \]
\[ w_1(3) = w_1(2) + (a(3) - w_1(2))p(2) = 3 + 0 = 3 \]

(no response)