Hebbian Learning

Topic 6

Note: lecture notes by Michael Negnevitsky (university of Tasmania), Bob Keller (Harvey Mudd College, CA) and Martin Hagan (University of Colorado) are used

Main idea: learning based on association between neurons

The main property of a neural network is an ability to learn from its environment, and to improve its performance through learning. So far we have considered **supervised** or **active learning** – learning with an external "teacher" or a supervisor who presents a training set to the network. But another type of learning also exists: **unsupervised learning**. In contrast to supervised learning, unsupervised or **self-organised learning** does not require an external teacher. During the training session, the neural network receives a number of different input patterns, discovers significant features in these patterns and learns how to classify input data into appropriate categories. Unsupervised learning tends to follow the neuro-biological organisation of the brain.

- Unsupervised learning algorithms aim to learn rapidly and can be used in real-time.

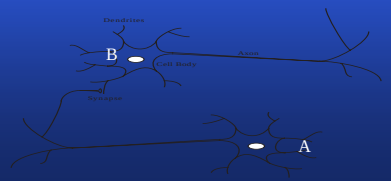
Hebbian learning

In 1949, Donald Hebb proposed one of the key ideas in biological learning, commonly known as **Hebb's Law**. Hebb's Law states that if neuron *i* is near enough to excite neuron *j* and repeatedly participates in its activation, the synaptic connection between these two neurons is strengthened and neuron *j* becomes more sensitive to stimuli from neuron *i*.

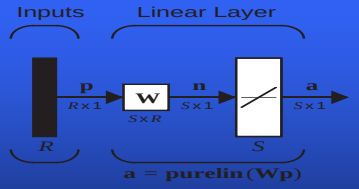
Hebb's Postulate

"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



Linear Associator



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

Training Set:

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

Hebb Rule

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

Postsynaptic Signal

Presynaptic Signal

Simplified Form:

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

Supervised Form:

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

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:

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

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

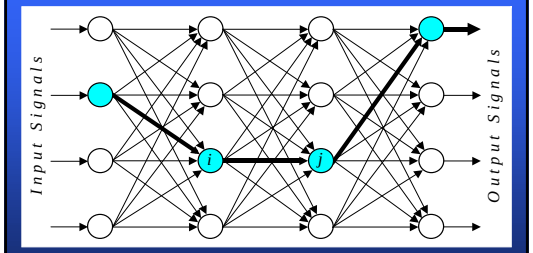
$$\mathbf{T} = \begin{bmatrix} \mathbf{t}_1 & \mathbf{t}_2 & \cdots & \mathbf{t}_Q \end{bmatrix}$$

Hebb's Law can be represented in the form of two rules:

1. **If two neurons on either side of a connection are activated synchronously, then the weight of that connection is increased.**
2. **If two neurons on either side of a connection are activated asynchronously, then the weight of that connection is decreased (added later)**

Hebb's Law provides the basis for learning without a teacher. Learning here is a **local phenomenon** occurring without feedback from the environment.

Hebbian learning in a neural network



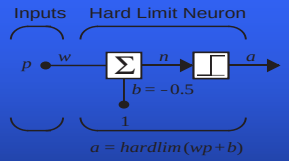
- Hebbian learning implies that weights can only increase. To resolve this problem, we might impose a limit on the growth of synaptic weights. It can be done by introducing a non-linear **forgetting factor** into Hebb's Law:

$$\Delta w_{ij}(p) = \alpha\ y_j(p)\ x_i(p) - \varphi\ y_j(p)\ w_{ij}(p)$$

where φ is the forgetting factor.

Forgetting factor usually falls in the interval between 0 and 1, typically between 0.01 and 0.1, to allow only a little "forgetting" while limiting the weight growth.

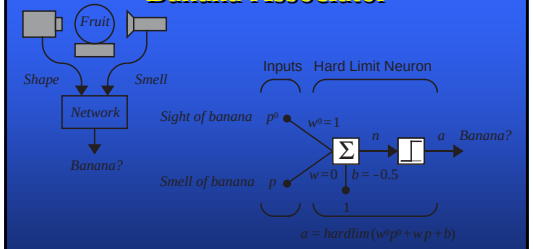
Simple Associative Network



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

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

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

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

Vector Form:

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

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):

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

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

Example

Second Iteration (sight works):

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

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

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

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

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

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

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

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):

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

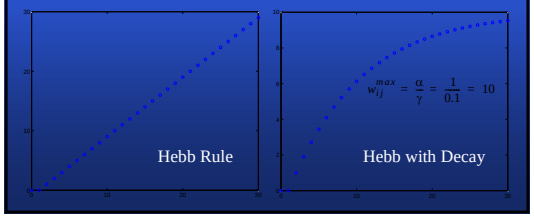
$$w(2) = w(1) + a(2)p(2) - 0.1w(1) = 0 + 1 \cdot 1 - 0.1(0) = 1$$

Example

Third Iteration (sight fails):

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

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



Problem of Hebb with Decay

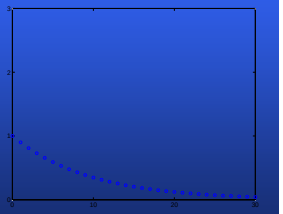
- Associations will decay away if stimuli are not occasionally presented.

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.

- Using Hebb's Law we can express the adjustment applied to the weight w_{ij} at iteration p in the following form:

$$\Delta w_{ij}(p) = F[y_j(p), x_i(p)]$$

- As a special case, we can represent Hebb's Law as follows:

$$\Delta w_{ij}(p) = \alpha \, y_j(p) \, x_i(p)$$

where α is the *learning rate* parameter.

This equation is referred to as the **activity product rule**.

Hebbian learning algorithm

Step 1: Initialisation.

Set initial synaptic weights and thresholds to small random values, say in an interval [0, 1].

Step 2: Activation.

Compute the neuron output at iteration p

$$y_j(p) = \sum_{i=1}^{n} x_i(p) \, w_{ij}(p) - \theta_j$$

where n is the number of neuron inputs, and θ_j is the threshold value of neuron j.

Step 3: Learning.

Update the weights in the network:

$$w_{ij}(p+1) = w_{ij}(p) + \Delta w_{ij}(p)$$

where $\Delta w_{ij}(p)$ is the weight correction at iteration p.

The weight correction is determined by the generalised activity product rule:

$$\Delta w_{ij}(p) = \varphi \, y_j(p)[\lambda \, x_i(p) - w_{ij}(p)]$$

Step 4: Iteration.

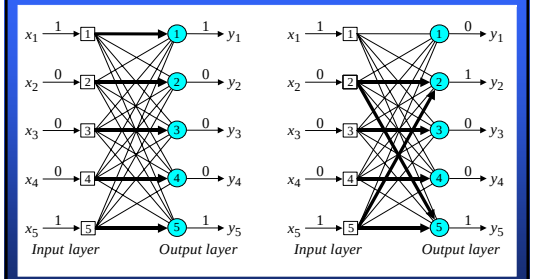
Increase iteration p by one, go back to Step 2.

Hebbian learning example

To illustrate Hebbian learning, consider a fully connected feedforward network with a single layer of five computation neurons. Each neuron is represented by a McCulloch and Pitts model with the sign activation function. The network is trained on the following set of input vectors:

$$\mathbf{X}_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \mathbf{X}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad \mathbf{X}_3 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad \mathbf{X}_4 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad \mathbf{X}_5 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Initial and final states of the network



Initial and final weight matrices

$$\begin{array}{c} \\ \text{Input layer} \end{array} \begin{array}{c} \begin{array}{ccccc} \text{Output layer} & & & & \\ 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array} \qquad \begin{array}{c} \\ \text{Input layer} \end{array} \begin{array}{c} \begin{array}{ccccc} \text{Output layer} & & & & \\ 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 2.0204 & 0 & 0 & 2.0204 \\ 0 & 0 & 1.0200 & 0 & 0 \\ 0 & 0 & 0 & 0.9996 & 0 \\ 0 & 2.0204 & 0 & 0 & 2.0204 \end{bmatrix} \end{array}$$

- A test input vector, or probe, is defined as

$$\mathbf{X} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

- When this probe is presented to the network, we obtain:

$$\mathbf{Y} = sign\left\{ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 2.0204 & 0 & 0 & 2.0204 \\ 0 & 0 & 1.0200 & 0 & 0 \\ 0 & 0 & 0 & 0.9996 & 0 \\ 0 & 2.0204 & 0 & 0 & 2.0204 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 0.4940 \\ 0.2661 \\ 0.0907 \\ 0.9478 \\ 0.0737 \end{bmatrix} \right\} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

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