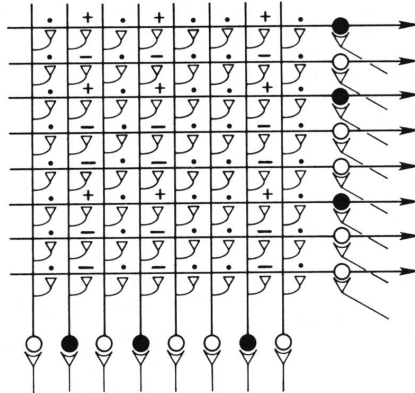


## Learning associations in connectionist networks

Association by *contiguity*, generalization by *similarity* (James, 1890)

- Represent items as patterns of activity, where *similarity* is reflected by overlap or correlation between patterns
- Represent *contiguity* as simultaneous presence of patterns over two groups of units (A and B)
- Adjust weights on connections between A and B so that the pattern on A tends to cause the corresponding pattern on B
- As a result, when the same or similar pattern is presented on A, it tends to produce the corresponding pattern on B (perhaps somewhat weakened/distorted)



If  $w_{ij} = 0$  initially, after a set of  $n$  training trials on patterns  $\mathbf{p}$  where  $\Delta w_{ij} = \epsilon a_i a_j$ ,

$$w_{ij} = \epsilon \sum_p a_i(\mathbf{p}) a_j(\mathbf{p})$$

Suppose  $a_i$  and  $a_j$  take on values of  $+1$  or  $-1$

- If  $a_i$  and  $a_j$  are *perfectly correlated* (always the same),  $a_i(\mathbf{p}) a_j(\mathbf{p}) = 1$ , so

$$w_{ij} = \epsilon n$$

- If  $a_i$  and  $a_j$  are *perfectly anticorrelated* (always different),  $a_i(\mathbf{p}) a_j(\mathbf{p}) = -1$ , so

$$w_{ij} = -\epsilon n$$

- If  $a_i$  and  $a_j$  are *uncorrelated* (as often different as same)

$$w_{ij} = \epsilon \left( \frac{n}{2} (+1) + \frac{n}{2} (-1) \right) = 0$$

- If  $a_i$  and  $a_j$  are *partially correlated* (e.g., 3/4 same and 1/4 different)

$$w_{ij} = \epsilon n \left( \frac{3}{4} (+1) + \frac{1}{4} (-1) \right) = \frac{1}{2} \epsilon n$$

- Thus  $w_{ij} \propto \text{correlation}(a_i, a_j)$

## Correlational learning: Hebb rule

### What Hebb actually said:

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

### The minimal version of the Hebb rule:

When there is a synapse between cell A and cell B, increment the strength of the synapse whenever A and B fire together (or in close succession).

### The minimal Hebb rule as implemented in a network:

$$\Delta w_{ij} = \epsilon a_i a_j$$

A statistical correlation is defined as

$$\frac{\sum_p (a_i(\mathbf{p}) - \overline{a_i(\mathbf{p})}) (a_j(\mathbf{p}) - \overline{a_j(\mathbf{p})})}{\sqrt{\left( \sum_p (a_i(\mathbf{p}) - \overline{a_i(\mathbf{p})})^2 \sum_p (a_j(\mathbf{p}) - \overline{a_j(\mathbf{p})})^2 \right)}}$$

where  $\overline{a_i(\mathbf{p})}$  is the mean of  $a_i(\mathbf{p})$ .

- Subtraction "takes out the mean"
- Denominator normalizes with respect to variation in  $a_i$  and  $a_j$

Hebb rule often includes "reference" values  $r_i$  and  $r_j$

$$\Delta w_{ij} = \epsilon (a_i - r_i) (a_j - r_j)$$

- Ordinarily do not use denominator
- Both issues go away with  $\pm 1$ , zero-mean patterns

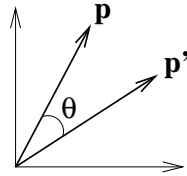
## Vector similarity: Dot product

- Inner (dot) product: Measure of *similarity* between two vectors (e.g.  $\mathbf{p}$  and  $\mathbf{p}'$ )

Let  $a_i(\mathbf{p})$  be the elements of activity pattern  $\mathbf{p}$

$$dp(\mathbf{p}, \mathbf{p}') = \mathbf{p} \cdot \mathbf{p}' = \sum_i a_i(\mathbf{p}) a_i(\mathbf{p}')$$

$$\cos \theta_{\mathbf{p}\mathbf{p}'} = \frac{\mathbf{p} \cdot \mathbf{p}'}{\|\mathbf{p}\| \|\mathbf{p}'\|}$$



- $\mathbf{p}$  and  $\mathbf{p}'$  are *orthogonal* if  $dp(\mathbf{p}, \mathbf{p}') = 0$
- Note that  $n_j = \sum_i a_i w_{ij} = dp(\mathbf{a}, \mathbf{w}_j)$

If test pattern  $\mathbf{p}'$  is orthogonal to all training patterns  $\mathbf{p}$ ,  $dp(\mathbf{p}', \mathbf{p}) = 0$  for all  $\mathbf{p}$ , so

$$a_j(\mathbf{p}') = \varepsilon \sum_p a_j(\mathbf{p}) dp(\mathbf{p}', \mathbf{p}) = \varepsilon \sum_p a_j(\mathbf{p}) 0 = 0$$

If all training patterns are orthogonal to each other (and, for simplicity,  $\varepsilon = 1$ )

- If  $\mathbf{p}'$  is one of the training patterns (say  $\mathbf{p}^*$ ), recall is perfect:

$$a_j(\mathbf{p}') = a_j(\mathbf{p}^*) dp(\mathbf{p}', \mathbf{p}^*) + \sum_{p \neq p^*} a_j(\mathbf{p}) dp(\mathbf{p}', \mathbf{p}) = a_j(\mathbf{p}^*) 1 + \sum_{p \neq p^*} a_j(\mathbf{p}) 0 = a_j(\mathbf{p}^*)$$

- If  $\mathbf{p}'$  is *similar* to only one training pattern ( $\mathbf{p}^*$ ) and orthogonal to the rest, the output is  $a_j(\mathbf{p}^*)$  scaled by the degree of similarity:

$$a_j(\mathbf{p}') = a_j(\mathbf{p}^*) dp(\mathbf{p}', \mathbf{p}^*) + \sum_{p \neq p^*} a_j(\mathbf{p}) dp(\mathbf{p}', \mathbf{p}) = a_j(\mathbf{p}^*) dp(\mathbf{p}', \mathbf{p}^*)$$

- In general, the output to any input pattern is a weighted combination of the outputs of all trained patterns, scaled by their similarity to the input.
  - If the combination agrees with  $a_j(\mathbf{p}')$ , this is **generalization**
  - If the combination disagrees with  $a_j(\mathbf{p}')$ , this is **interference**

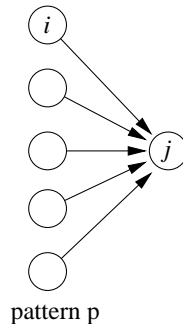
## How training patterns influence unit activations

If  $w_{ij} = 0$  initially, after training on a set of patterns  $\mathbf{p}$  using  $\Delta w_{ij} = \varepsilon a_i a_j$ ,

$$w_{ij} = \varepsilon \sum_p a_i(\mathbf{p}) a_j(\mathbf{p})$$

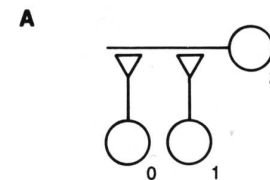
After training, response of linear unit to test pattern  $\mathbf{p}'$ :

$$\begin{aligned} a_j(\mathbf{p}') &= \sum_i a_i(\mathbf{p}') w_{ij} \\ &= \sum_i a_i(\mathbf{p}') \left( \varepsilon \sum_p a_i(\mathbf{p}) a_j(\mathbf{p}) \right) \\ &= \varepsilon \sum_p a_j(\mathbf{p}) \sum_i a_i(\mathbf{p}') a_i(\mathbf{p}) \\ &= \varepsilon \sum_p a_j(\mathbf{p}) dp(\mathbf{p}', \mathbf{p}) \\ &= \varepsilon \sum_p a_j(\mathbf{p}) \text{similarity}(\mathbf{p}', \mathbf{p}) \end{aligned}$$



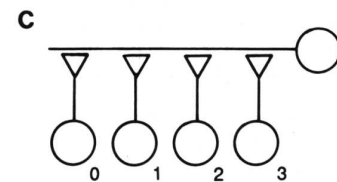
Response of output unit  $j$  to pattern  $\mathbf{p}'$  is *combination* of its response to known patterns  $\mathbf{p}$ , weighted by their *similarity* to  $\mathbf{p}'$

## Limitations of Hebbian learning



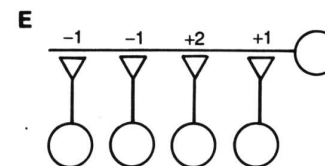
**B**

| Input |   | Output |
|-------|---|--------|
| 0     | 1 | 2      |
| +     | + | +      |
| +     | - | +      |
| -     | + | -      |
| -     | - | -      |



**D**

| Input |   |   |   | Output |
|-------|---|---|---|--------|
| 0     | 1 | 2 | 3 | 4      |
| +     | - | + | - | +      |
| +     | + | + | + | +      |
| +     | + | + | - | -      |
| +     | - | - | + | -      |



## Error-correcting learning: Delta rule

Change weights so as to reduce difference between actual output ( $a_j$ ) and **target** output (denoted  $t_j$ )

$$\Delta w_{ij} = \varepsilon(t_j - a_j) a_i$$

- “Delta”: difference between output and target
  - Also called Widrow-Hoff rule, LMS (least mean squared)
  - Related to perceptron convergence procedure (Rosenblatt)
- Hebb rule:  $\Delta w_{ij} = \varepsilon t_j a_i$  (where  $t_j$  is activation “clamped” on the output unit)
- Similar to correlation with *error*

Weight changes focus on *predictive differences*

- Hebb/correlational learning depends on predictive *similarities*

## Effects of training on response to input patterns

Calculated in terms of *changes* to activations caused by learning on single pattern  $p$ :

$$\begin{aligned} \Delta a_j(p') &= \sum_i a_i(p') \Delta w_{ij} \\ &= \sum_i a_i(p') \varepsilon(t_j(p) - a_j(p)) a_i(p) \\ &= \varepsilon(t_j(p) - a_j(p)) \sum_i a_i(p') a_i(p) \\ &= \varepsilon(t_j(p) - a_j(p)) \text{dp}(\mathbf{p}', \mathbf{p}) \end{aligned}$$

- If  $\mathbf{p}$  and  $\mathbf{p}'$  are orthogonal, training on  $\mathbf{p}$  will have no effect on  $\mathbf{p}'$
- If  $\mathbf{p}$  and  $\mathbf{p}'$  are not orthogonal, training on  $\mathbf{p}$  will affect performance on  $\mathbf{p}'$  (weighted by similarity) which may be good (generalization) or bad (interference)

## Learning on orthogonal patterns: Delta = Hebb

$$\text{Delta rule: } \Delta w_{ij} = \varepsilon(t_j - a_j) a_i$$

For first pattern  $p = 0$ ,  $w_{ij} = 0$  so  $a_j(0) = n_j(0) = 0$ , and

$$w_{ij} = \Delta w_{ij} = \varepsilon(t_j(0) - 0) = t_j(0) a_i(0)$$

For next pattern  $p = 1$ ,  $a_j(1) = \sum_i a_i(1) w_{ij} = \sum_i a_i(1) t_j(0) a_i(0) = t_j(0) \sum_i a_i(1) a_i(0)$ .

Since patterns 0 and 1 are orthogonal,  $\sum_i a_i(1) a_i(0) = 0$ , so  $a_j(1) = 0$ . Thus

$$\begin{aligned} \Delta w_{ij} &= t_j(1) a_i(1) \\ w_{ij} &= t_j(0) a_i(0) + t_j(1) a_i(1) \end{aligned}$$

In fact,  $a_j(p) = 0$  for the first presentation of each training pattern  $p$ , so at the end:

$$w_{ij} = \varepsilon \sum_p (t_j(p) - a_j(p)) a_i(p) = \varepsilon \sum_p t_j(p) a_i(p)$$

This is just Hebbian learning using targets  $t_j$  as output activations ( $a_j$ ).

Note that the Delta rule is inherently *multi-pass*

- In general,  $a_j \neq 0$  on subsequent presentations of each training pattern
- Weight changes due to one pattern affect error on others

## Delta rule as gradient descent in error (linear units)

$$a_j = \sum_i a_i w_{ij}$$


$$\text{Error } E = \frac{1}{2} \sum_j (t_j - a_j)^2$$

$$w_{ij} \searrow \quad t_j \searrow \\ a_i \rightarrow \quad a_j \rightarrow E$$

$$\text{Gradient descent: } \Delta w_{ij} = -\varepsilon \frac{\partial E}{\partial w_{ij}}$$

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial a_j} \frac{\partial a_j}{\partial w_{ij}} \quad (\text{Chain rule})$$

$$= -(t_j - a_j) a_i$$

$$\Delta w_{ij} = -\varepsilon \frac{\partial E}{\partial w_{ij}} = \varepsilon(t_j - a_j) a_i$$

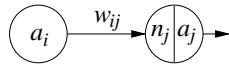
$$= \text{Delta rule}$$

## Delta rule as gradient descent in error (sigmoid units)

$$n_j = \sum_i a_i w_{ij}$$

$$a_j = \frac{1}{1 + \exp(-n_j)}$$

$$\text{Error } E = \frac{1}{2} \sum_j (t_j - a_j)^2$$

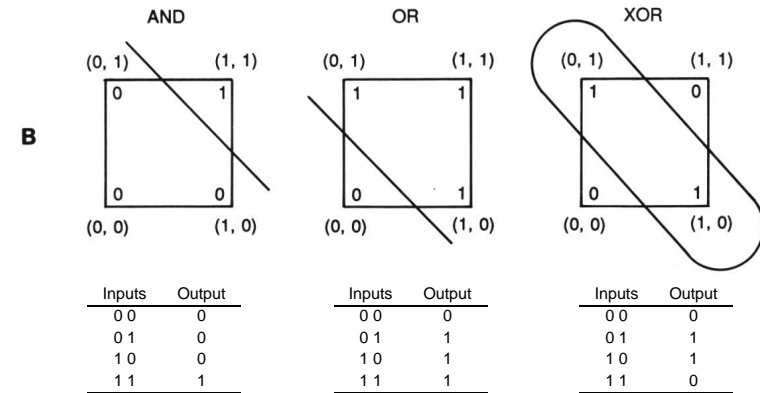
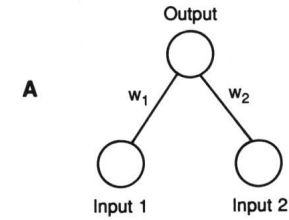


$$w_{ij} \xrightarrow{a_i} n_j \rightarrow a_j \xrightarrow{t_j} E$$

$$\text{Gradient descent: } \Delta w_{ij} = -\varepsilon \frac{\partial E}{\partial w_{ij}}$$

$$\begin{aligned} \frac{\partial E}{\partial w_{ij}} &= \frac{\partial E}{\partial a_j} \frac{da_j}{dn_j} \frac{\partial n_j}{\partial w_{ij}} \\ &= -(t_j - a_j) a_j (1 - a_j) a_i \end{aligned}$$

$$\Delta w_{ij} = -\varepsilon \frac{\partial E}{\partial w_{ij}} = \varepsilon (t_j - a_j) a_j (1 - a_j) a_i$$



## When does the Delta rule succeed or fail?

Delta rule is *optimal*

- Will find a set of weight that produces zero error *if such a set exists*

Guaranteed to succeed if input patterns are **linearly independent**

- No pattern can be created by recombining the others
- i.e., there is *something unique* about each pattern (cf. Hebb rule: no similarity (orthogonal))

Succeed at binary classification of outputs: **Linear separability**

- Weights define a plane (line for two input units) through activation (state) space
- Must be possible to position a plane such that all patterns requiring  $n_j < 0$  are on one side and all patterns requiring  $n_j > 0$  are on the other side
- AND** and **OR** are linearly separable but **XOR** is not

**XOR** can be made linearly separable by adding a new "input"

- Corresponds to a third dimension in state space
- Intermediate ("hidden") units can learn what new value is necessary

| Inputs | Output |
|--------|--------|
| 00     | 0      |
| 01     | 1      |
| 10     | 1      |
| 11     | 0      |

