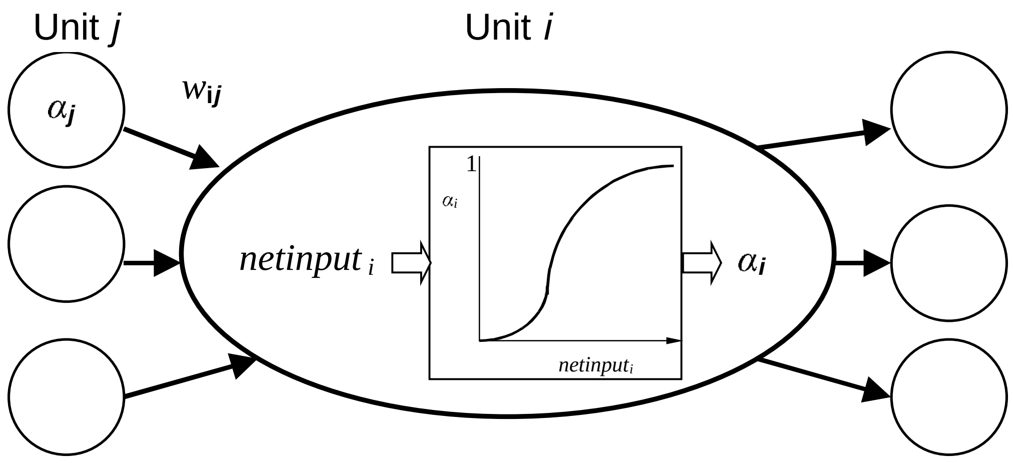
Knowledge representation in Artificial Neural Networks

- Knowledge representation is *distributed* across many processing units



Integrate input from previous layer

Transform *netinput* to activity level (α_i)

Transmit activity level to units in next layer

- Computations take place in *parallel* across these distributed representations

- Conclusions are reached on the basis of a consensus of many calculations rather than depending on any particular one.

The representation of knowledge in neural networks is distributed

- *Traditional models in AI usually assume a local representation of knowledge:*

<u>knowledge about different things is stored in different, independent locations</u>

Examples:
➨Familiar *information storage systems*, such as dictionaries, telephone directories, computer discs, use local representation. Each discrete piece of information is stored separately.

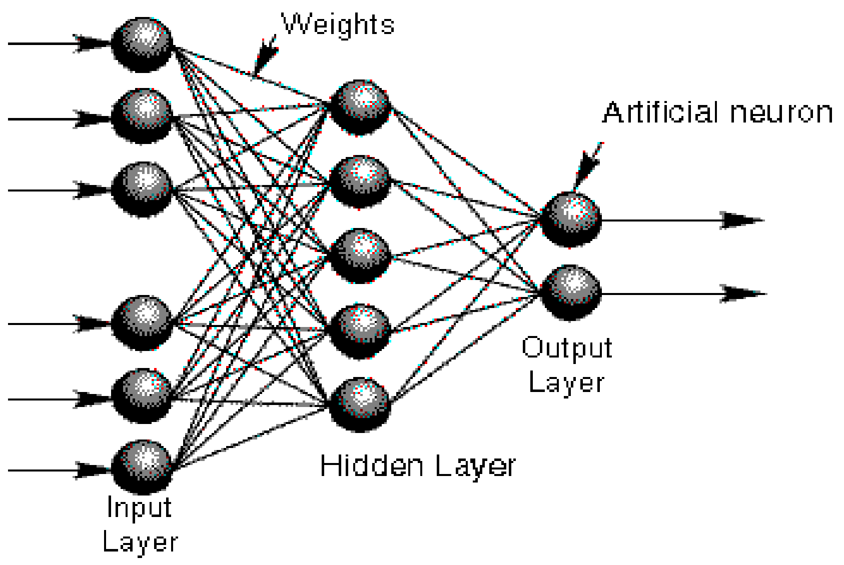
➨In a traditional model of *reading aloud*, information about how to pronounce the letter string DOG is stored in one place and information about how to pronounce the string CAT in another. In general, these models contain a lexicon, a store of specific information about particular words with which the reader is familiar, and some other mechanisms, such as a set of rules for pronunciation, to allow pronunciation of novel letter strings.

- *In neural networks, information storage is not local, it is distributed:*

<u>There is no one place where a particular piece of knowledge can be located.</u>

Example:

➨In a *neural network that learns to read aloud*, any input, such as the letter string DOG, excites units and connections all over the network.



Learning takes place by changing the weights of the connections leading to all output units, which have an incorrect level of activity. The knowledge of how to pronounce the input DOG is distributed across many different connections in different parts of the network. <u>It is the sum total effect of all these connections, which produces the pronunciation, not any single one of them.</u>

- All the knowledge that the network contains is superimposed on the same set of connections.
- The connections which contain the network's knowledge about how to pronounce DOG are the same as those with the knowledge about how to pronounce any other letter string.
- There are no familiar information storage systems which use distributed encoding.

Pattern association

1. Architecture and properties of a specific kind of network, called *pattern associator*.

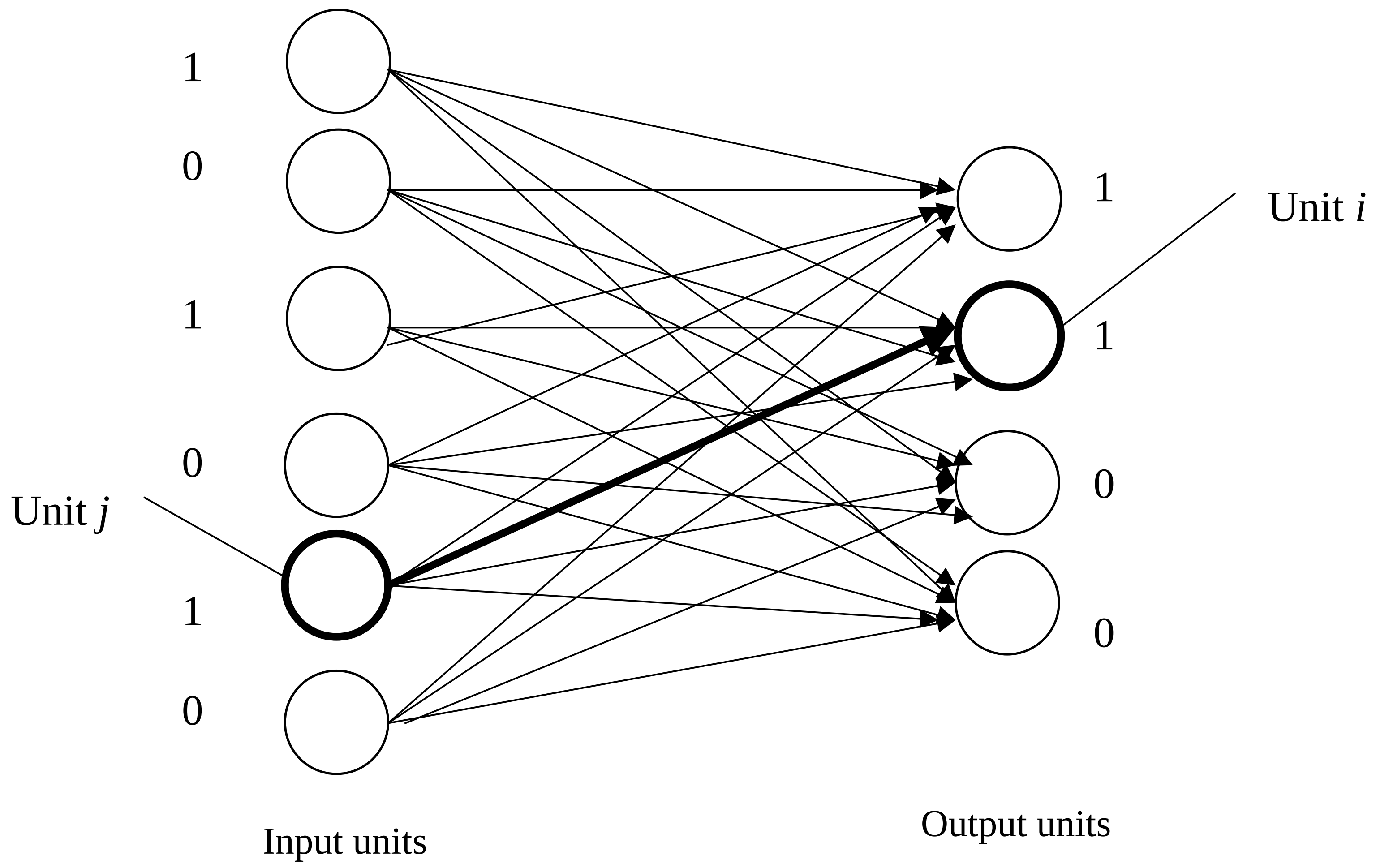
2. The operation of a particular learning rule, which is called the *Hebb rule*.

3. *Training phase*: during training a pattern associator is presented with pairs of patterns.

4. *Operation phase*: after training a pattern associator will recall one of the patterns at the output when the other is presented at the input. It can also respond to novel inputs, generalising from its experience with similar patterns.

5. *Main properties*: Tolerance to noisy inputs and resistance to internal damage. It is capable of extracting the central tendency or prototype from a set of similar examples.

A network for performing pattern association



The Hebb rule

IF there is activity on input neuron *j* when the neuron *i* is active THEN the strength of the connection , w_{ij}, between *j* and *i* is increased.

$$w_{ij}^{new} = w_{ij}^{old} + \Delta w_{ij}$$

$$\Delta w_{ij} = \varepsilon \alpha_i \alpha_j$$

Example

Weight matrix before learning:

$$\begin{array}{c|cccccc}
 & j=1\ldots6 & & & & & \\
i=1,\ldots4 & 1 & 0 & 1 & 0 & 1 & 0 \\
\hline
1\leftarrow & 0 & 0 & 0 & 0 & 0 & 0 \\
1\leftarrow & 0 & 0 & 0 & 0 & 0 & 0 \\
0\leftarrow & 0 & 0 & 0 & 0 & 0 & 0 \\
0\leftarrow & 0 & 0 & 0 & 0 & 0 & 0
\end{array}$$

Weight matrix after one learning trial:

$$\begin{array}{c|cccccc}
 & j=1\ldots6 & & & & & \\
i=1,\ldots4 & 1 & 0 & 1 & 0 & 1 & 0 \\
\hline
1\leftarrow & 1 & 0 & 1 & 0 & 1 & 0 \\
1\leftarrow & 1 & 0 & 1 & 0 & 1 & 0 \\
0\leftarrow & 0 & 0 & 0 & 0 & 0 & 0 \\
0\leftarrow & 0 & 0 & 0 & 0 & 0 & 0
\end{array}$$

Recall from a Hebb trained network

The net input to the output neuron *i* is

$$netinput_i = \sum_j \alpha_j w_{ij}$$

Input pattern: (1 0 1 0 1 0)

netinput₁: (1 x 1 + 0 x 0 + 1 x 1 + 0 x 0 + 1 x 1 + 0 x 0) = 3

1	0	1	0	1	0	
1	0	1	0	1	0	$\rightarrow$3
1	0	1	0	1	0	$\rightarrow$3
0	0	0	0	0	0	$\rightarrow$0
0	0	0	0	0	0	$\rightarrow$0

Activation function: ?

Threshold : ?

Learning different associations on the same connections

What happens if we try to store different associations on the same set of weights?

Can they still be recalled correctly or will they interfere?

Example

Weight matrix after one learning trial:

$$\begin{array}{lcccccc}
 & j=1\ldots6 & & & & & \\
 & 1 & 1 & 0 & 0 & 0 & 1 \\
i=1,\ldots4 & & & & & & \\
0\leftarrow & 0 & 0 & 0 & 0 & 0 & 0 \\
1\leftarrow & 1 & 1 & 0 & 0 & 0 & 1 \\
0\leftarrow & 0 & 0 & 0 & 0 & 0 & 0 \\
1\leftarrow & 1 & 1 & 0 & 0 & 0 & 1
\end{array}$$

Superimposing the weights acquired during learning to associate the two different pairs of patterns (i.e. summing the two weight matrices) produces:

$$\begin{array}{cccccc}
1 & 0 & 1 & 0 & 1 & 0 \\
2 & 1 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 1
\end{array}$$

Crucial test

1	1	0	0	0	1

1	0	1	0	1	0	$\rightarrow$1
2	1	1	0	1	1	$\rightarrow$4
0	0	0	0	0	0	$\rightarrow$0
1	1	0	0	0	1	$\rightarrow$3

Activation function: binary with output threshold set to 2

Output pattern: ?

1	0	1	0	1	0

1	0	1	0	1	0	$\rightarrow$3
2	1	1	0	1	1	$\rightarrow$4
0	0	0	0	0	0	$\rightarrow$0
1	1	0	0	0	1	$\rightarrow$1

Activation function: binary with output threshold set to 2

Output pattern: ?

Remarks

- Accurate recall of different associations from a pattern associator trained with the Hebb rule is possible even when the associations are stored on the same connections.
- The recalled patterns do not exactly match the original patterns. Even with only two patterns some interference between the patterns is already apparent.
- The differences have been removed by the threshold setting of the activation function
- There is a limit to the number of associations which can be stored in such network before interference between the patterns becomes a problem.
- What happens when the number of patterns is increased?

Generalisation

What happens when the associator is presented with the pattern

(1 1 0 1 0 0)

which is similar, but not identical, to the pattern

(1 1 0 0 0 1)

which it learnt before?

1	1	0	1	0	0	
1	0	1	0	1	0	$\rightarrow$1
2	1	1	0	1	1	$\rightarrow$3
0	0	0	0	0	0	$\rightarrow$0
1	1	0	0	0	1	$\rightarrow$2

Activation function: binary with output threshold set to 2

Output pattern: (0 1 0 1)

<u>The network has treated the new pattern as a noisy version of a pattern it already knew, and produced the response it learnt to that.</u>

Fault tolerance

What happens if some of the connections on neuron *i* are damaged after learning?

1	1	0	0	0	1	
1	0	1	0	1	0	→1
2	1	1	0	X	1	→4
0	0	0	0	0	0	→0
1	X	0	0	0	1	→2

Activation function: binary with output threshold set to 2

Output pattern: (0 1 0 1)

Minor damage will cause a small change in response to many inputs, rather than a total loss of some memories and no effect on others. This called *graceful degradation*