Pattern Association Memory

Background Reading

Rolls, E.T. and Treves, A. (1998). *Neural Networks and Brain Function*. Oxford University Press.

- Chapter 1: Introduction to general concepts
- Chapter 2: Main chapter on pattern association memory
- Chapter 7: Pattern association in the brain – amygdala and orbitofrontal cortex
- Appendix A3: Further technical details of pattern association

Overview

- Introduction to pattern association and neural networks
- Properties of pattern association networks employing distributed representations
- Linear separability (a potential problem)
- Pattern association in the brain

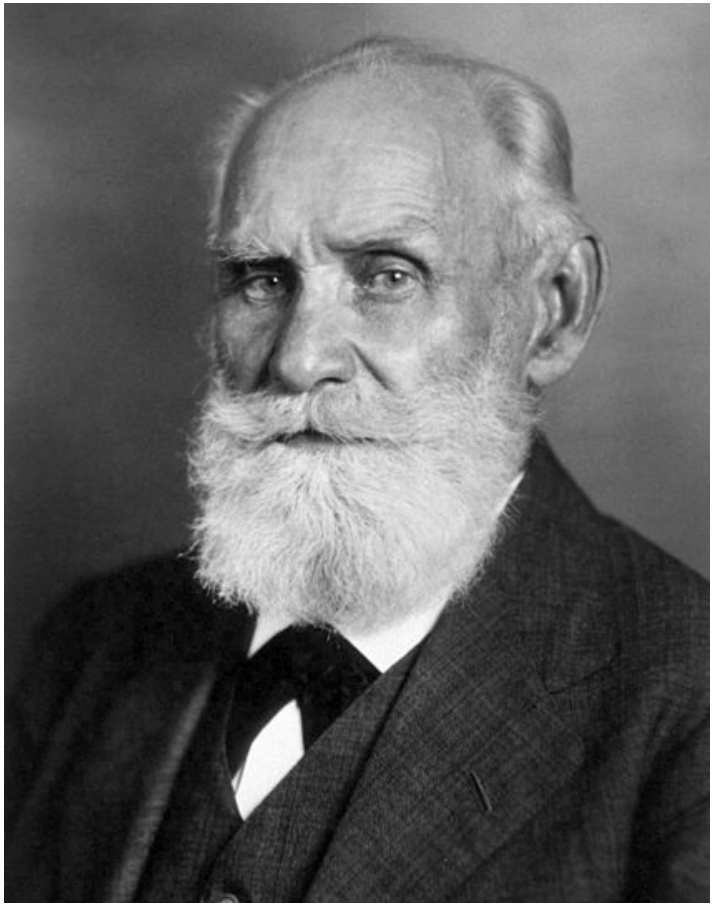
What is Pattern Association?

- A basic operation of most nervous systems is to learn to associate a first stimulus with a second stimulus that occurs at the same time
- For instance, the first stimulus might be the sight of food and the second stimulus might be the taste of food
- Through associative learning, the sight of the food enables the taste of the food to be retrieved: food that tastes nice will be preferentially selected in the future

What is Pattern Association?

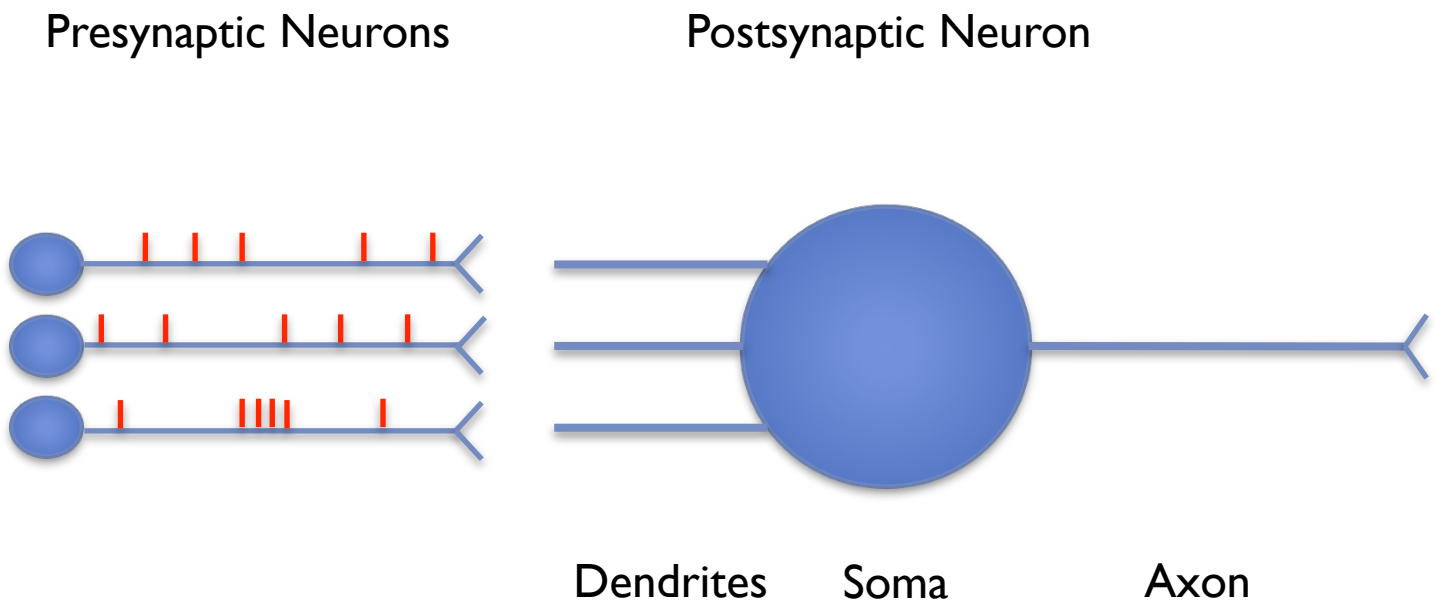
- Pattern association can also explain classical conditioning: learning to associate a conditioned stimulus with a conditioned response

- Pavlov's Dogs: through conditioning (pattern association) a ringing bell alone elicits salivation – originally a response to the sight of food

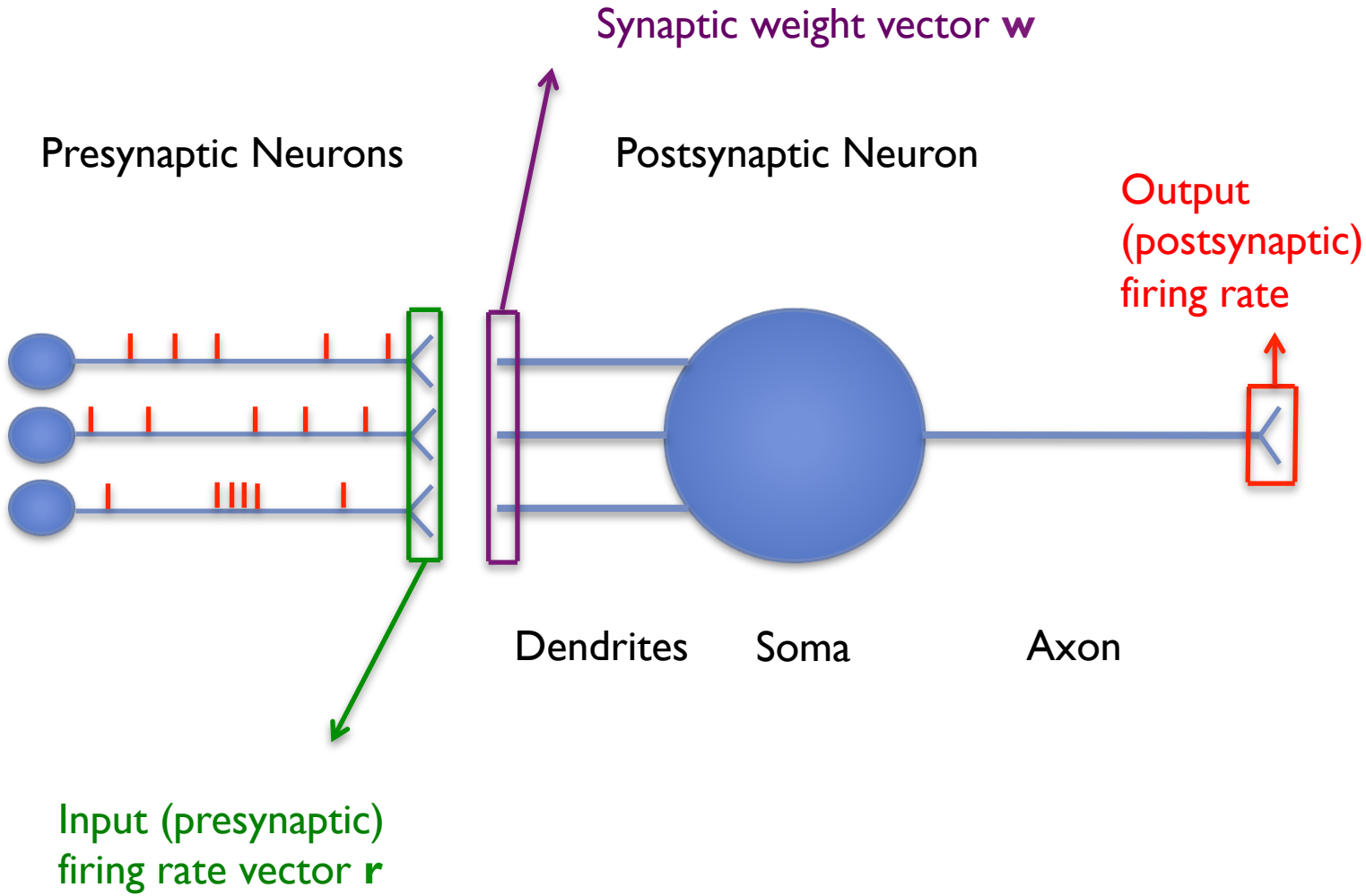


Ivan Pavlov

Modelling a Single Neuron



Modelling a Single Neuron



Modelling a Single Neuron

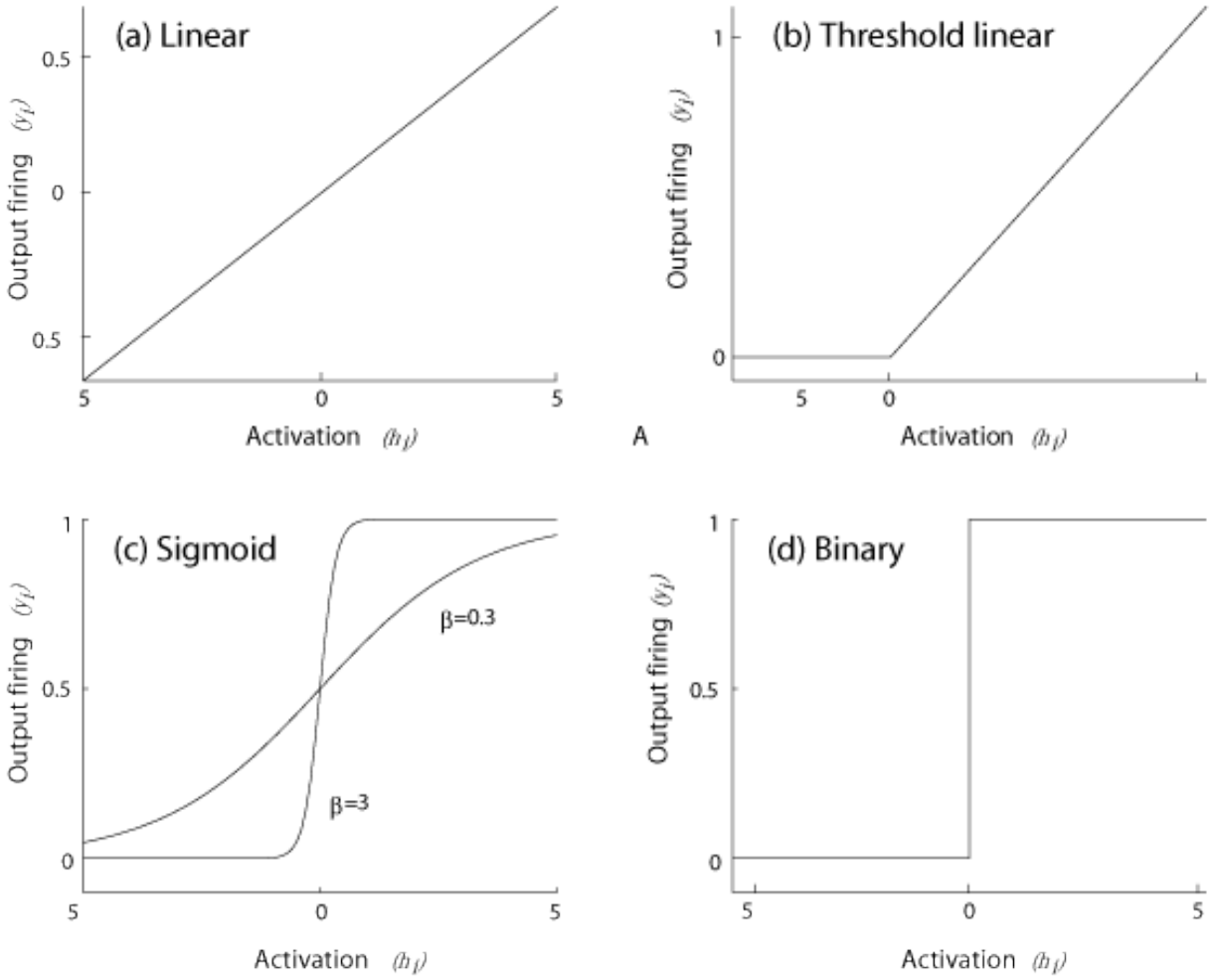
The activation level h_i of a postsynaptic neuron is calculated as the dot product between the synaptic weight vector $\mathbf{w}_i$ of that neuron and the presynaptic firing rate vector $\mathbf{r}_j$

$$h_i = \mathbf{w}_i \cdot \mathbf{r}$$

The firing rate r_i of a postsynaptic neuron is calculated by passing the activation level h_i through an activation (transfer) function

$$r_i = f(h_i)$$

Activation Functions



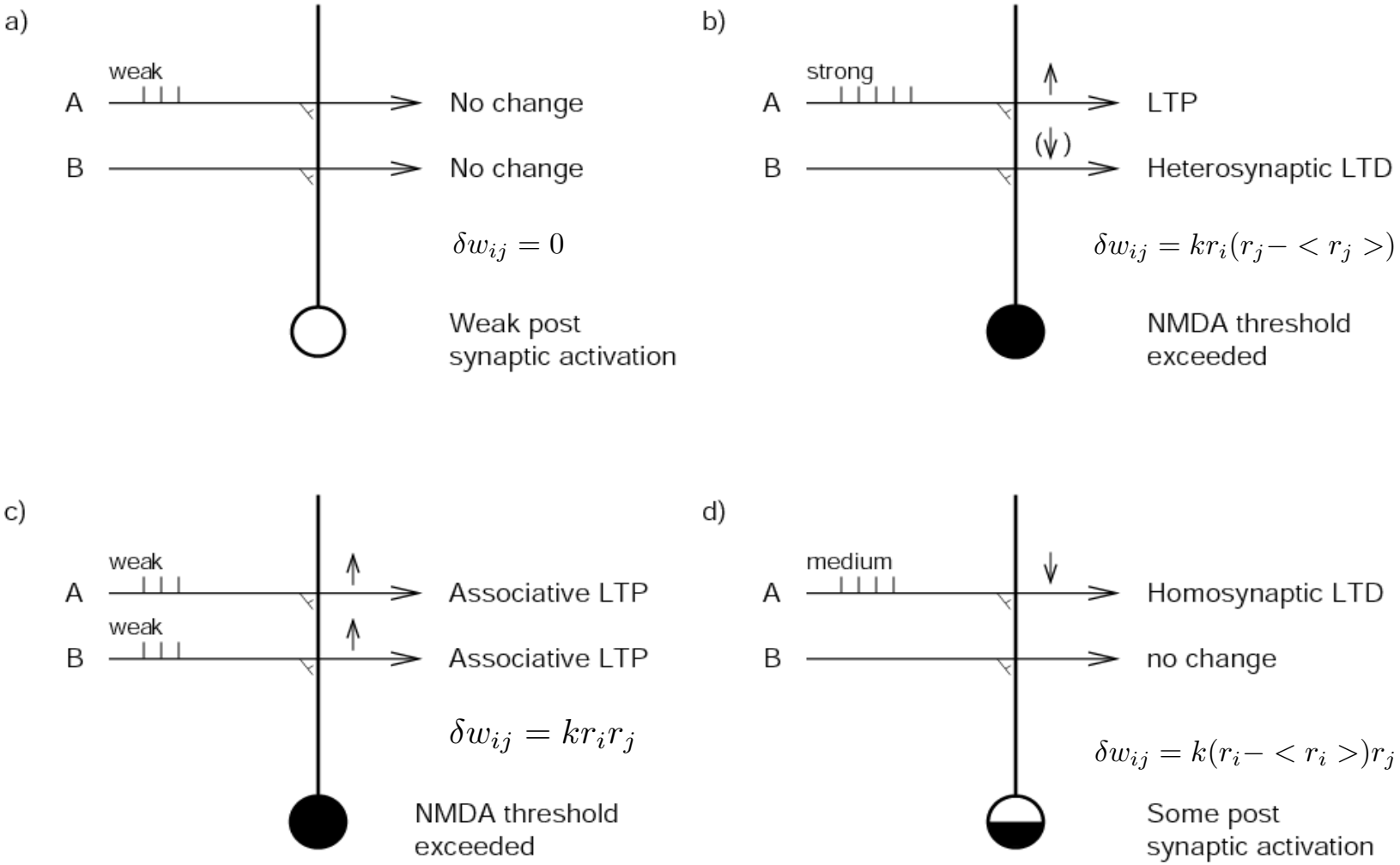
Associative Learning (The Hebb Rule)

- The change in strength of an individual synapse is related to the strengths of the presynaptic and postsynaptic firing rates

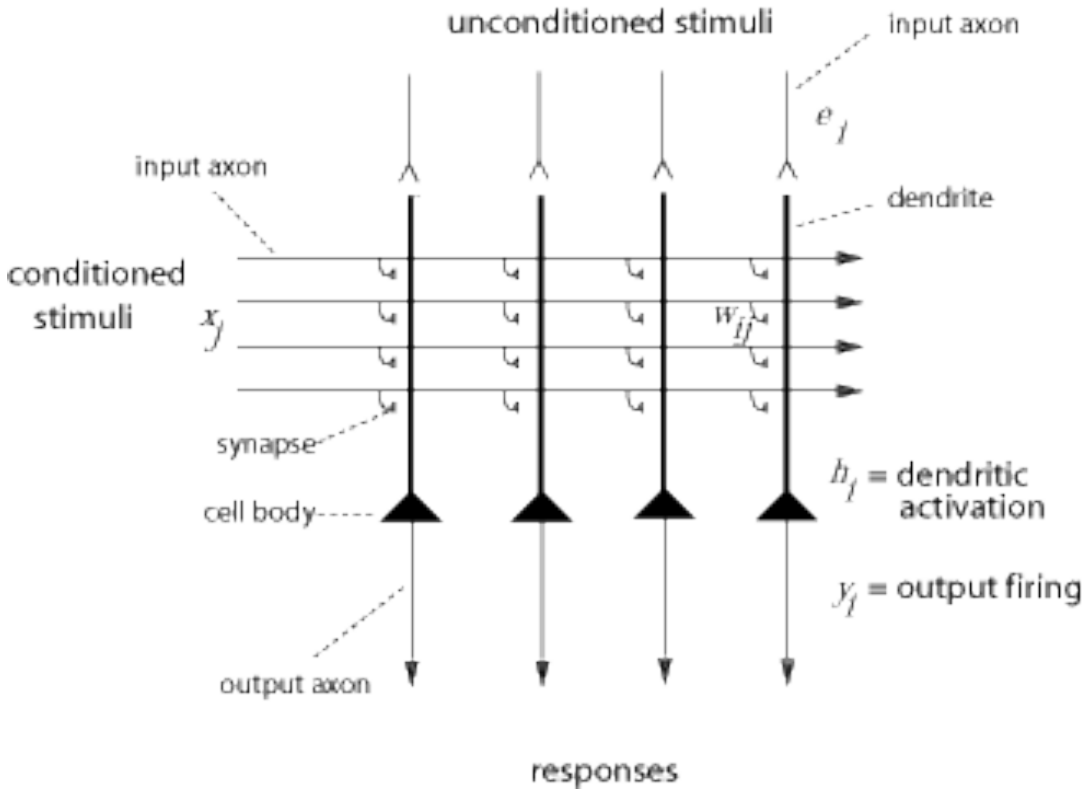
$$\delta w_{ij} = k r_i r_j$$

Donald Hebb

Learning at Synapses: LTP and LTD



Network Architecture



Learning a Pattern Association

Unconditioned Stimulus

	1	1	0	0
1	0	0	0	0
0	0	0	0	0
1	0	0	0	0
0	0	0	0	0
1	0	0	0	0
0	0	0	0	0

Conditioned Stimulus

synapses

- Start with synaptic weights set to zero
- Apply Conditioned Stimulus (CS) to input cells and Unconditioned Stimulus (UCS) to output cells

Learning a Pattern Association

Unconditioned Stimulus

	1	1	0	0
Conditioned Stimulus: 1	1	1	0	0
Conditioned Stimulus: 0	0	0	0	0
Conditioned Stimulus: 1	1	1	0	0
Conditioned Stimulus: 0	0	0	0	0
Conditioned Stimulus: 1	1	1	0	0
Conditioned Stimulus: 0	0	0	0	0

synapses

- Apply the Hebb Rule

$$\delta \mathbf{w}_i = k y_i \mathbf{x}$$

Weight Vectors After Learning

- The UCS forces firing of the output cells
- When a CS is applied to the input cells then learning will occur (provided the output cells are still firing)
- The weight vector on each output neuron will be

$$\mathbf{w}_i = k y_i \mathbf{x}$$

- This is a scaled version of the input (CS) firing rate vector: output neurons that are active during learning become responsive to the presence of the CS

Recall After Learning

During recall, the activation of each output neuron *i* is

$$\begin{aligned} h_i &= \mathbf{w}_i \cdot \mathbf{x} \\ &= (k y_i \mathbf{x}) \cdot \mathbf{x} \\ &= (k\mathbf{x} \cdot \mathbf{x}) y_i \end{aligned}$$

The vector of output activations is

$$\mathbf{h} = (k\mathbf{x} \cdot \mathbf{x})\mathbf{y}$$

When the learned CS vector **x** is applied at recall, the vector of output cell activations **h** is equal to a scaled version of the original UCS output firing rate vector **y**

Recalling a Pattern Association

Firing Rate	1	1	0	0
Activation	3	3	0	0

Conditioned Stimulus					synapses
1	1	1	0	0	
0	0	0	0	0	
1	1	1	0	0	
0	0	0	0	0	
1	1	1	0	0	
0	0	0	0	0	

- Calculate activations of each output neuron

$$h_i = \mathbf{w}_i \cdot \mathbf{x}$$

- Apply binary threshold transfer function to calculate firing rate (set to 1 if activation greater than 2, 0 otherwise)

Training With A Second Pair of Patterns

Unconditioned Stimulus

	0	1	0	1
1	1	2	0	1
1	0	1	0	1
0	1	1	0	0
0	0	0	0	0
0	1	1	0	0
1	0	1	0	1

Conditioned Stimulus

synapses

- Apply the Hebb Rule

$$\delta \mathbf{w}_i = k y_i \mathbf{x}$$

Training With A Second Pair of Patterns

Firing Rate	0	1	0	1
Activation	1	4	0	3

Conditioned Stimulus	synapses			
1	1	2	0	1
1	0	1	0	1
0	1	1	0	0
0	0	0	0	0
0	1	1	0	0
1	0	1	0	1

- Calculate activations of each output neuron

$$h_i = \mathbf{w}_i \cdot \mathbf{x}$$

- Apply binary threshold transfer function to calculate firing rate (set to 1 if activation greater than 2, 0 otherwise)

Local vs. Distributed Representations

- Local representation
  - Information that a stimulus occurred is carried by a single neuron
  - The number of different stimuli that can be represented by a set of neurons is **low** (at maximum *n* stimuli with *n* neurons)
  - Not fault tolerant to noise or damage

Local vs. Distributed Representations

- Distributed representation
  - Information that a stimulus occurred is signalled by the activity of the whole set of neurons
  - The number of different stimuli that can be represented is **high**
  - The network is fault tolerant to noise or damage

Distributed Representations

- A distributed representation can be either:
  - Fully Distributed, in which approximately half of the neurons are active for any one stimulus or event
  - Sparse Distributed, in which less than half of the neurons are active for any one stimulus or event

Generalisation

- The dot product of two vectors provides a measure of the similarity between the two vectors
- Previously unseen input firing rate vectors, if they are similar to an experienced input firing rate vector, will have a high dot product with the stored weight vectors
- Distributed representations allow for generalisation (this is **not** possible with a local representation)

Generalisation

Firing Rate	0	1	0	1
Activation	1	3	0	2

Conditioned Stimulus					synapses
1	1	2	0	1	
1	0	1	0	1	
0	1	1	0	0	
1	0	0	0	0	
0	1	1	0	0	
0	0	1	0	1	

- Calculate activations of each output neuron

$$h_i = \mathbf{w}_i \cdot \mathbf{x}$$

- Apply binary threshold transfer function to calculate firing rate (set to 1 if activation greater than 2, 0 otherwise)

Graceful Degradation/Fault Tolerance

- The dot product, as a measure of similarity between two vectors, also allows for a degree of resistance to error
- Within reasonable limits, a noisy version of a previously experienced stimulus, or damage to a network, will still produce a high dot product between the input firing rate vector and the stored weight vector
- We often experience noisy versions of a stimulus, e.g. faces under different lighting conditions

Graceful Degradation/Fault Tolerance

Firing Rate	0	1	0	1
Activation	1	4	0	2

Conditioned Stimulus					synapses
1	1	2	0	1	
1	0	1	0	x	
0	1	1	0	0	
0	0	0	0	0	
0	1	x	0	0	
1	0	1	0	1	

- Calculate activations of each output neuron

$$h_i = \mathbf{w}_i \cdot \mathbf{x}$$

- Apply binary threshold transfer function to calculate firing rate (set to 1 if activation greater than 2, 0 otherwise)

Extraction of Central Tendency or Prototype

- When a set of similar CS vectors $\mathbf{x}_j$ are paired with the same UCS vector $\mathbf{y}$, the weight vector $\mathbf{w}_i$ points towards the average of the CS vectors

- The weight vector $\mathbf{w}_i$ extracts the central tendency or prototype of the CS vectors

Speed of the Network

- Recall is fast: the activation vector **h** can be accumulated in one or two time constants of the dendrite (10-20 ms)

- Learning is also fast ("one-shot"): a single pairing of the CS and UCS is enough to enable the association to be learned

Capacity and Interference

In linear networks the firing rate of an output neuron is the same as its activation level

$$r_i = h_i$$

Assume we wish to train, using the Hebb rule, two different CS input vectors **x** and **z** to map to two different output firing rate vectors $\mathbf{y}^x$ and $\mathbf{y}^z$

Capacity and Interference

Training with CS input vector **x** adds a component to the weight vector $\mathbf{w}_i$ of each output neuron *i* that is a scaled version of the input pattern **x**

$$\delta \mathbf{w}_i = k y_i^x \mathbf{x}$$

Capacity and Interference

Training with CS input vector **z** adds an additional component to the weight vector $\mathbf{w}_i$ of each output neuron *i* that is a scaled version of the input pattern **z**

$$\delta \mathbf{w}_i = k y_i^z \mathbf{z}$$

Thus, after training, the weight vector of each output cell is given by

$$\mathbf{w}_i = k y_i^x \mathbf{x} + k y_i^z \mathbf{z}$$

Capacity and Interference

If, after training, we present the CS input vector **x** to the network then the activation of each output cell *i* is given by

$$\begin{aligned} h_i &= \mathbf{w}_i \cdot \mathbf{x} \\ &= (ky_i^x \mathbf{x} + ky_i^z \mathbf{z}) \cdot \mathbf{x} \\ &= (ky_i^x \mathbf{x} \cdot \mathbf{x}) + (ky_i^z \mathbf{z} \cdot \mathbf{x}) \end{aligned}$$

Capacity and Interference

Considering the vector of output cell activations **h**

$$\mathbf{h} = k(\mathbf{x} \cdot \mathbf{x})\mathbf{y}^x + k(\mathbf{z} \cdot \mathbf{x})\mathbf{y}^z$$

The first term is a scaled version of the UCS vector $\mathbf{y}^x$ associated with CS vector **x**

The second term is a scaled version of the UCS vector $\mathbf{y}^z$ associated with CS vector **z**: this is interference!

It is only if CS input vectors **x** and **z** are orthogonal, i.e. dot product of 0, that there will not be interference

Capacity and Interference

Interference between different patterns limits the storage capacity of pattern association networks

It is possible to reduce interference between non-orthogonal patterns by training with a modified Hebb rule

$$\delta w_{ij} = k y_i (x_j - x)$$

where x is a constant (often set to the average value of x_j)

The modified Hebb rule allows for heterosynaptic long-term depression, which will decrease synaptic strengths and reduce interference

With C input elements up to C input vectors can be learned by the network

Capacity in Non-Linear Neurons

- Non-linear neurons have non-linear activation functions, e.g. threshold, sigmoid
- The non-linearity results in clustering of input patterns
  - Similar input patterns are mapped close together in terms of output vectors
  - Dissimilar input patterns are mapped to different output vectors – reduces interference

Capacity in Non-Linear Neurons

- With fully distributed representations, the capacity is of the order *C*, where *C* is the number of input elements x_j per output neuron

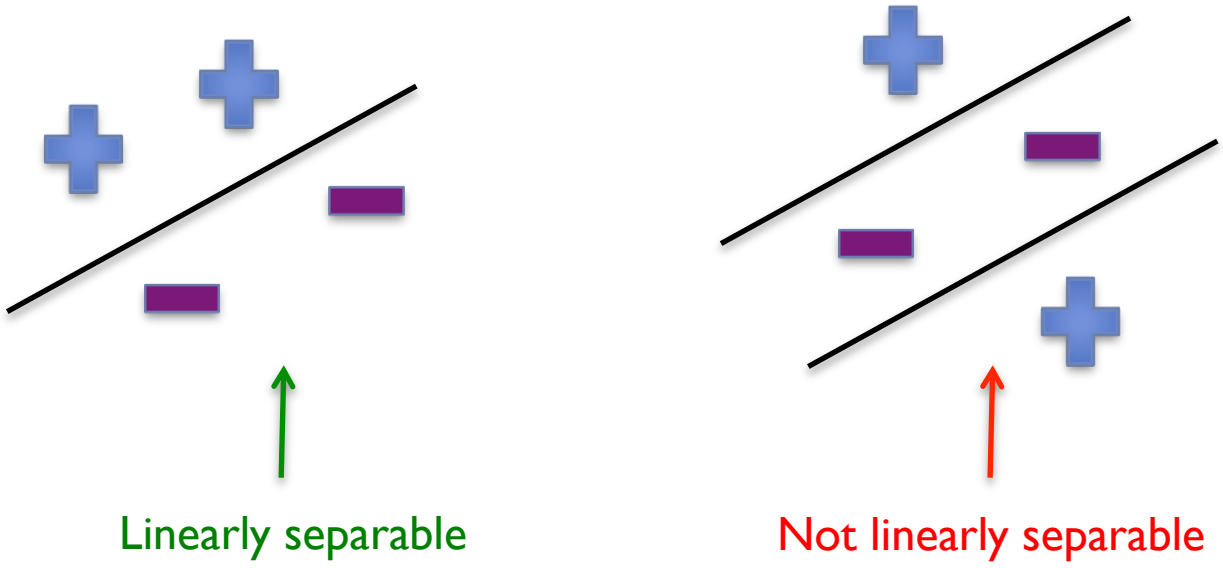
- With sparse distributed representations, many more patterns can be stored. Capacity is

$$p \approx \frac{C}{[a \log(1/a)]}$$

where *a* is the sparseness (proportion of output units firing)

Linear Separability (A Potential Problem)

- Two sets of points in *N*-dimensional space are linearly separable if they can divided into two classes by an *N-1*-dimensional hyperplane:



Linear Separability (A Potential Problem)

- Linear separability relates to classification
- Is there a criterion by which we can divide the data points into two classes?
- If so, then the data are linearly separable and we can perform the classification

Linear Separability and Pattern Association

- If the data are linearly separable, then a pattern association network *can* learn the classification
- If the data are not linearly separable, then a pattern association network *cannot* learn the classification
- Expansion recoding, performed by a pre-processing competitive network, enables data that are not linearly separable to be classified by a pattern association network - more on this in the lecture on competitive networks

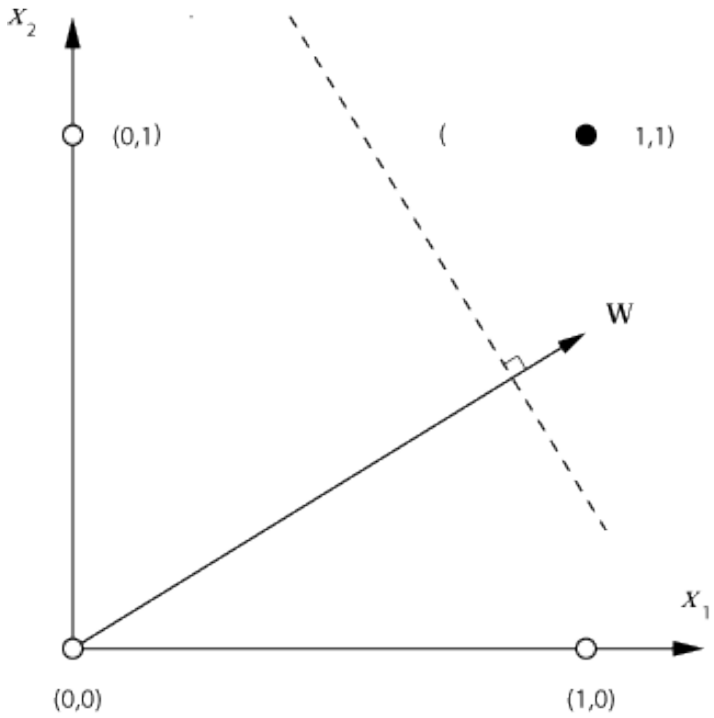
Linear Separability in Action

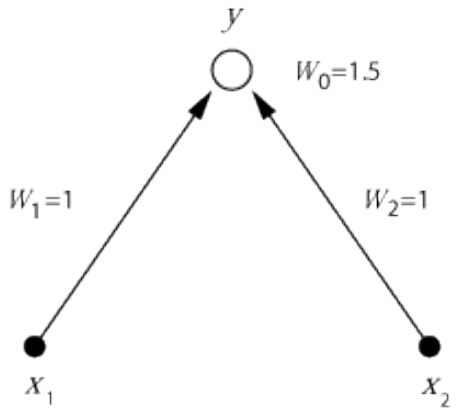
- The logical AND operator

A	B	Output
1	1	1
1	0	0
0	1	0
0	0	0

Linear Separability in Action

- The logical AND operator





A pattern association network can learn this, i.e. find the correct weight vector to partition the data points into two classes

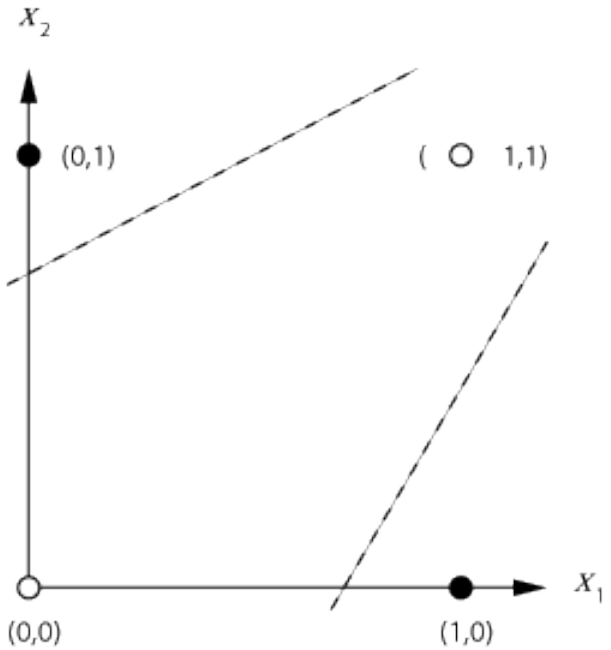
Data That Are Not Linearly Separable

- The logical XOR operator

A	B	Output
1	1	0
1	0	1
0	1	1
0	0	0

Data That Are Not Linearly Separable

- The logical XOR operator



A pattern association network cannot learn this, i.e. there is no weight vector that will partition the data points into two classes

Summary So Far

- Pattern association networks associate a stimulus with a second stimulus occurring at the same time
- This is an example of unsupervised associative learning

Summary So Far

- If a distributed representation is employed then pattern association networks exhibit:
  - Generalisation
  - Fault tolerance/graceful degradation
  - Extraction of central tendency
  - High capacity
- N.B. The above properties also hold for any neural network that employs a distributed representation

Summary So Far

- Pattern association networks cannot learn the correct solution when the data are not linearly separable

- In this case, a pattern association network will not learn to correctly classify the data points

Pattern Association in the Brain

- Amygdala and Orbitofrontal cortex
  - Visual-to-taste learning
  - Stimulus-reinforcement learning
- Cortico-cortical back projections
  - Re-instating a memory in cortex during recall

Amygdala and Orbitofrontal Cortex

- Neurons in the amygdala and orbitofrontal cortex may learn to associate a visual stimulus (CS) with taste (UCS)

- This is an example of stimulus-reinforcement learning, i.e. avoid food that tastes bad

Amygdala and Orbitofrontal Cortex

- Both the amygdala and orbitofrontal cortex receive afferent connections from the insula (primary taste cortex)

- The amygdala and orbitofrontal cortex also receive afferent connections from inferior temporal visual cortex

Amygdala and Orbitofrontal Cortex

- Lesions to the amygdala produce a deficit in learning to associate stimuli with rewards or punishments

- Primates with amygdala lesions will eat food they had previously rejected (Weiskrantz, 1956)

Amygdala and Orbitofrontal Cortex

- Lesions to orbitofrontal cortex affect tasks in which the association between a stimulus and a reward or punishment must be extinguished or reversed

- Primates with orbitofrontal cortex lesions will persist with behavioural responses to a stimulus long after that stimulus has ceased to be rewarded (Jones & Mishkin, 1972)

Cortico-Cortical Backprojections

- Hippocampal damage generally produces anterograde amnesia rather than retrograde amnesia (e.g. patient H.M.)
- This suggests longer-term memories are stored in cortex
- The correct firing patterns need to be re-instantiated in cortex during recall of the memory

Cortico-Cortical Backprojections

- Backprojections from cortex to cortex are hypothesised to produce the reinstatement of the memory during recall (McClelland, McNaughton & O'Reilly, 1995; Rolls, 1996)

- This would be a case of simple pattern association between a pattern of firing in one region of cortex and a pattern of firing in another region of cortex