3041 Neural computation: Models of brain function

- **Course organisers - contact details**

Dr Caswell Barry: e-mail: caswell.barry@ucl.ac.uk

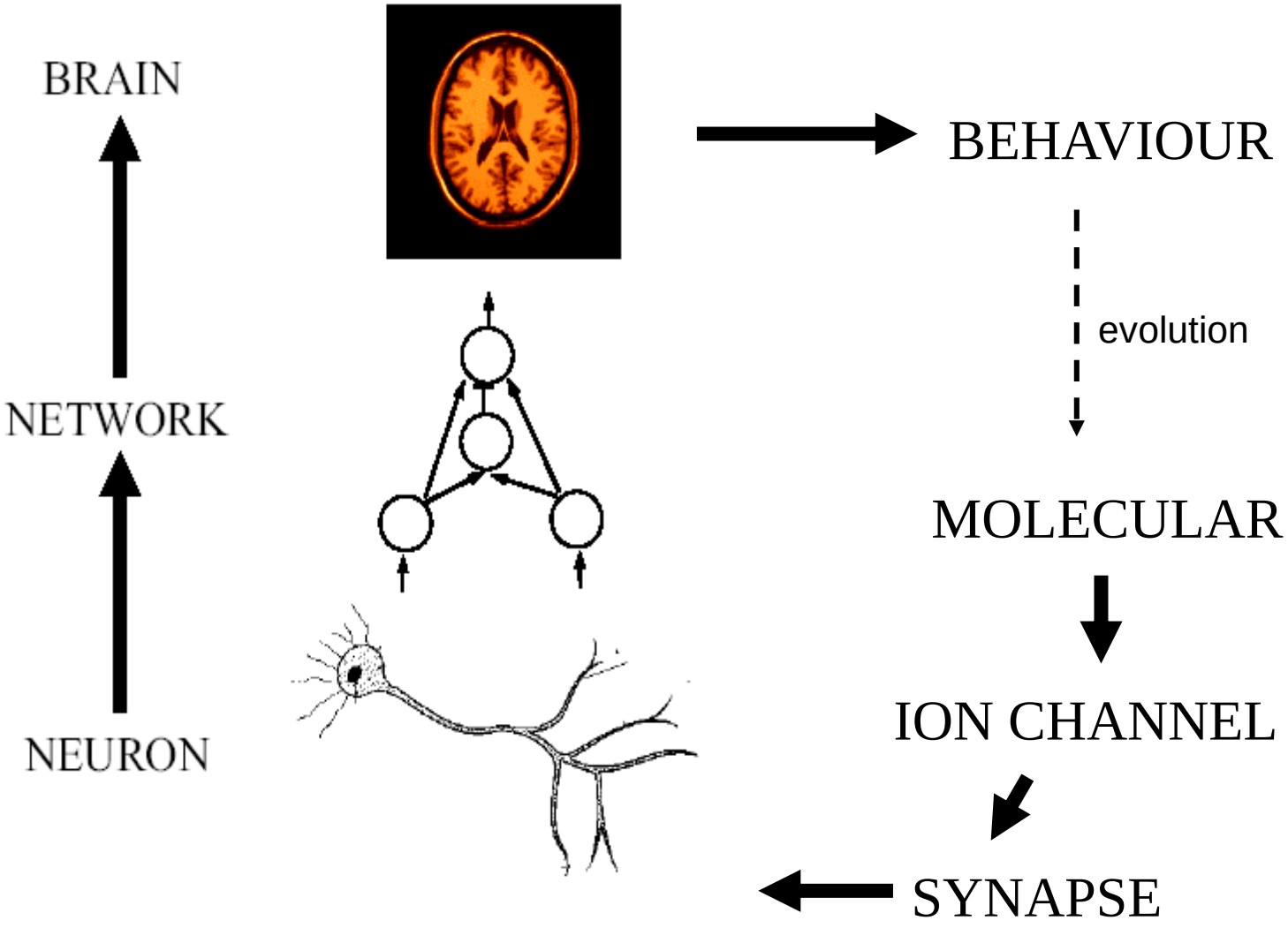
Prof Neil Burgess: e-mail: n.burgess@ucl.ac.uk

www.ucl.ac.uk/icn/people/space-and-memory/space-leader/neil-teaching

C41 – Half unit. Aims

- To introduce the consideration of neurons and synapses in terms of their computational properties, and interpretation of their action in terms of information processing.
- To introduce the analysis of an animal's ability to learn, remember or act in terms of the action of neurons and synapses within the animal's nervous system.
- To understand several examples of how the action of individual neurons and synapses in various parts of the central nervous system contribute to the learning, memory or behaviour of an organism.

1) **Levels of neurophysiological description**



2) ***Artificial* neural networks vs models of brain function**

Schedule 2017

Day	Time	Subject	Lecturer	Venue
13 Oct	10:00 – 11:00	Introduction to artificial neural networks & unsupervised learning.	Prof. Neil Burgess	Chadwick Building G07
18 Oct	11:00 – 12:00	Intro to artificial neural networks & unsupervised learning, cont.	Prof. Neil Burgess	Medawar Building G02 Watson LT
	12:00 – 13:00	Intro to artificial neural networks & unsupervised learning, cont	Prof. Neil Burgess	
20 Oct	10:00 – 11:00	Computational properties of individual neurons	Prof David Attwell	Chadwick Building G07
25 Oct	11:00 – 12:00	Artificial neural networks, feedback & simple supervised learning.	Prof. Neil Burgess	Medawar Building G02 Watson LT
	12:00 – 13:00	More advanced learning algorithms in artificial neural networks.	Prof. Neil Burgess	
27 Oct	10:00 – 11:00	More advanced learning algorithms in artificial neural networks, cont.	Prof. Neil Burgess	Chadwick Building G07
1 Nov	11:00 – 12:00	The hippocampus and spatial representation.	Dr Andrej Bicanski	Medawar Building G02 Watson LT
	12:00 – 13:00	Path integration, continuous attractors and grid cells.	Dr Daniel Bush	
3 Nov	10:00 – 11:00	Hippocampal and striatal navigation.	Dr Caswell Barry	Chadwick Building G07
		Reading Week		
15 Nov	11:00 – 12:00	Hippocampus and associative memory	Dr Andrej Bicanski	Medawar Building G02 Watson LT
	12:00 – 13:00	Hippocampus and associative memory	Dr Andrej Bicanski	
17 Nov	10:00 – 11:00	Spatial processing in the spine and motor cortex.	Dr Caswell Barry	Chadwick Building G07
22 Nov	11:00 – 12:00	Reinforcement learning.	Prof. Neil Burgess	Medawar Building G02 Watson LT
	12:00 – 13:00	Reinforcement learning, cont.	Prof. Neil Burgess	
24 Nov	10:00 – 11:00	Models of prefrontal cortex.	Dr Sam Gilbert	Chadwick Building G07
29 Nov	11:00 – 12:00	Learning, performing and remembering serially ordered actions.	Dr Caswell Barry	Medawar Building G02 Watson LT
	12:00 – 13:00	Neural bases of sensory decision making.	Prof Peter Latham	
1 Dec	10:00 – 11:00	NO LECTURE		
6 Dec	11:00 – 12:00	Filtering and normalization in sensory systems.	Prof Matteo Carandini	Medawar Building G02 Watson LT
	12:00 – 13:00	Theories of the cerebellum.	Dr Peter Gilbert	
8 Dec	10:00 – 11:00	Temporal processing in audition and olfaction.	Dr Caswell Barry	Chadwick Building G07
13 Dec	11:00 – 12:00	Computing with spike timing and delays; course review.	Prof. Neil Burgess	TBA

General reading list

General: Fundamentals of Computational Neuroscience by Thomas Trappenberg (OUP, 2002)

Artificial Neural Networks:

1. An Introduction to Neural Networks, James A. Anderson (MIT Press, 1995);
2. An Introduction to Neural Networks, Kevin Gurney (UCL Press, 1997);
3. Parallel Distributed Processing I: Foundations, Rumelhart & McClelland (MIT Press, 1986).
4. Parallel Distributed Processing II: Psychological and Biological Models (MIT Press, 1986).
5. Neural Networks for Control Miller W, Sutton R, Werbos P, (MIT Press, 1995)
6. Perceptrons Minsky M & Papert S (MIT Press 1969).
7. Genetic programming… Koza JR (MIT press, 1992).
8. Self-Organisation and Associative Memory. Kohonen T (Springer Verlag, 1989).

Biological neural networks:

9. The synaptic organisation of the brain. Shepard GM (Oxford University Press, 1979).
10. The computational brain. Churchland PS and Sejnowski TJ (MIT press, 1994)
11. The computing neuron. Durbin R, Miall C and Mitchison G (Addison Wesley, 1989).

Models of brain systems/ systems neuroscience:

12. The handbook of brain theory and neural networks. Arbib MA (ed) (MIT Press 1995)
13. The cognitive neurosciences. Gazzaniga MS (ed) (MIT Press 1995)
14. The hippocampal and parietal foundations of spatial cognition. Burgess N, Jeffery KJ and O'Keefe J (eds) (OUP 1999).

Computational Neuroscience (v. mathematical)

15. Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems Peter Dayan Larry F Abbott (MIT, 2001).
16. Intro to the Theory of Neural Computation. Hertz, Krogh & Palmer (Addison Wesley, 91).

Introduction to artificial neural networks & unsupervised learning

AIMS

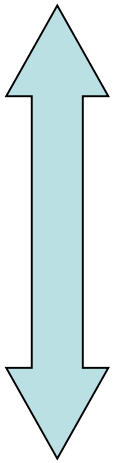
1. Understand simple mathematical models of how a neuron's firing rate depends on the firing rates of the neurons with synaptic connections to it.
2. Describe how Hebbian learning rules relate change in synaptic weights to the firing rates of the pre- and post-synaptic neurons.
3. Describe how application of these rules can lead to self-organisation in artificial neural networks.
4. Relate self-organisation in artificial neural networks to organisation of the brain, such as in topographic maps.

READING

1. Books 1,2,8.
2. Rumelhart DE & Zipser D (1986) `Feature discovery by competitive learning', in: Rumelhart D E and McClelland J L (Eds.) Parallel Distributed Processing 1 151-193 MIT Press.
3. Sharp P E (1991) `Computer simulation of hippocampal place cells', Psychobiology 19 103-115.
4. Kohonen T (1982) `Self-organised formation of topologically correct feature maps' Biological Cybernetics 43 59-69.
5. Udin S B, Fawcett J W (1988) `Formation of topographic maps' Annual Review of Neuroscience 11 289-327

Modelling the function of a neuron: levels of description

Very detailed

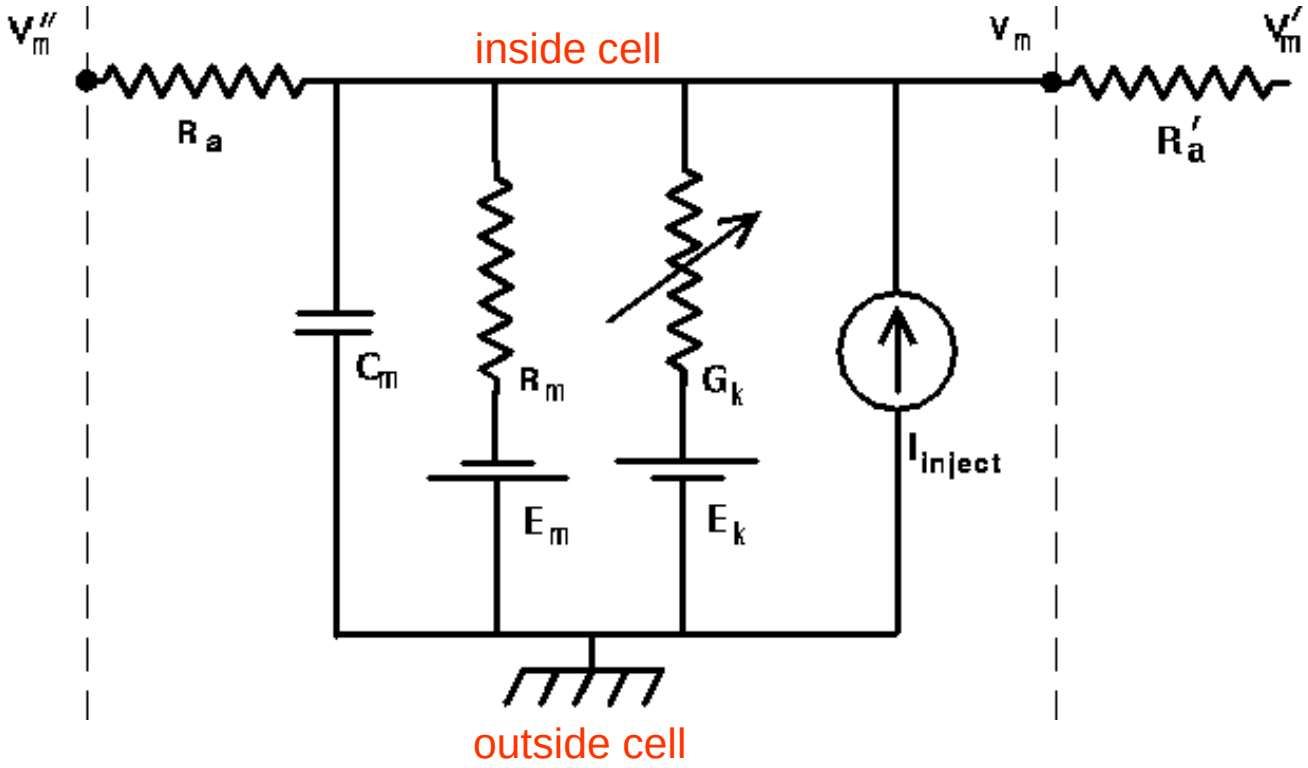


no detail

- full compartmental models
- leaky integrate-and-fire models
- integrate-and-fire models
- standard artificial neuron
- threshold logic unit

Keep it simple: there's going to be lots of them..

Compartmental models. Each compartment is simple:



V_m: membrane potential inside the compartment (relative to "ground" outside cell).

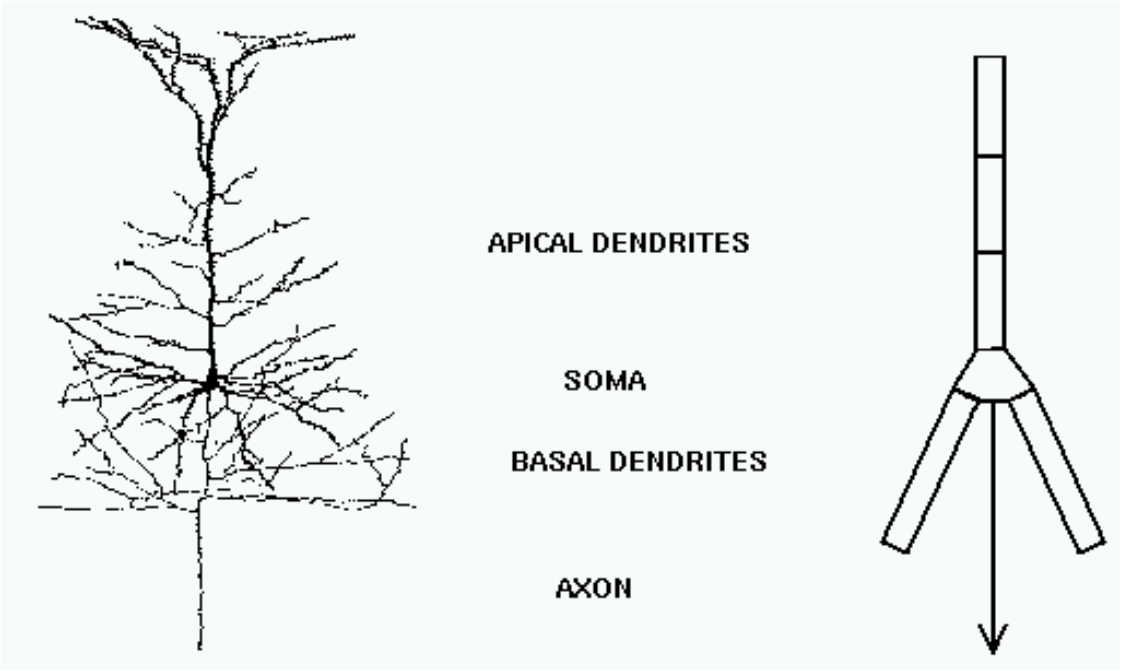
C_m: membrane capacitance - charged or discharged as ions flow in or out of the compartment (changing V_m) from adjacent compartments (V_m' and V_m'', axial resistances R_a and R_a') or through the membrane.

R_m: leakage resistance & equilibrium potential E_m represent passive channels (rate and direction of current flow depends on size V_m versus E_m).

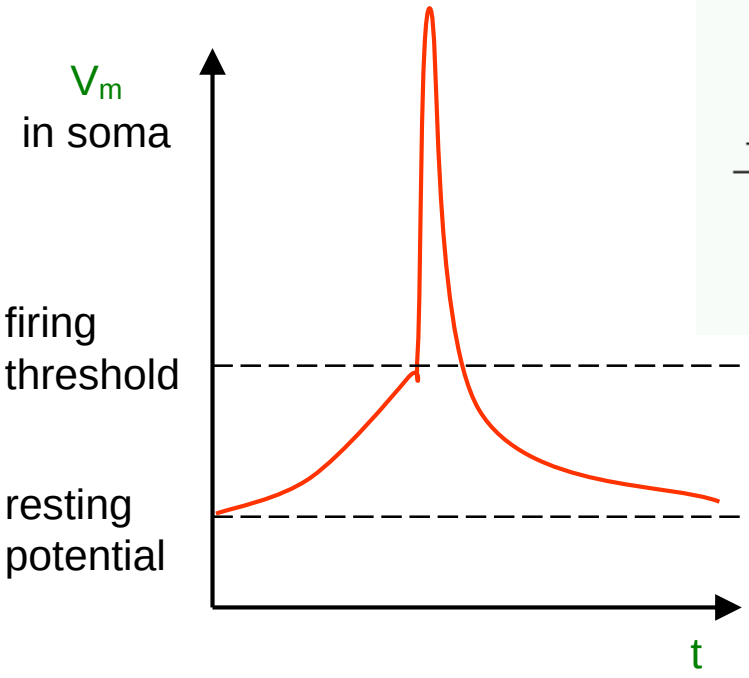
G_k: variable conductances (1/resistance) specific to particular ions. Each has its own equilibrium potential E_k and may vary with V_m (**active** channels).

Compartmental models, cont.

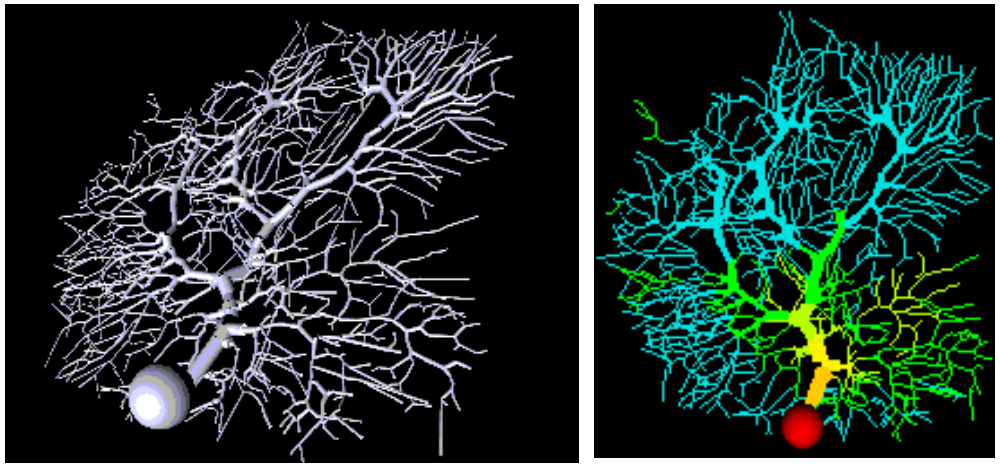
Model cell with small no. of compartments



action potential (‘spike’)



or large no. of compartments..



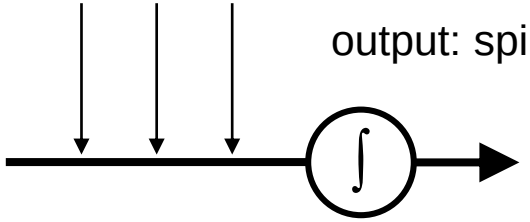
Computer simulation of a Purkinje cell, color represents membrane potential.

Integrate-and-fire models

inputs current pulses:

output: spikes

No physical structure:

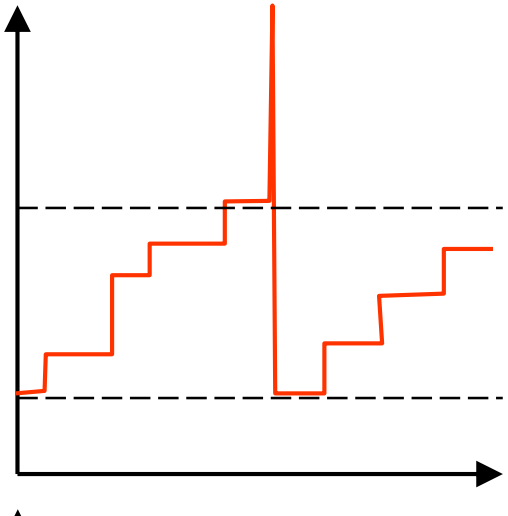


temporal integration

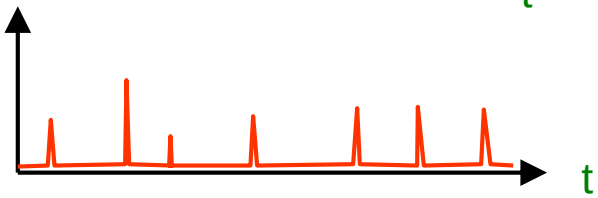
membrane potential (V)

firing threshold (T)

resting potential (V_r)



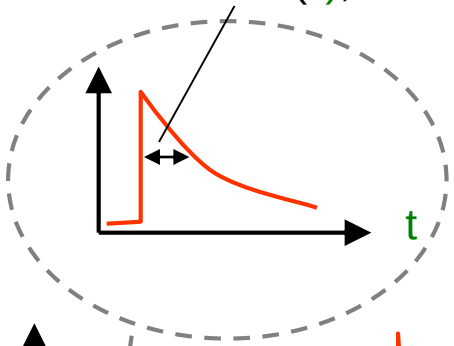
Input current (I)

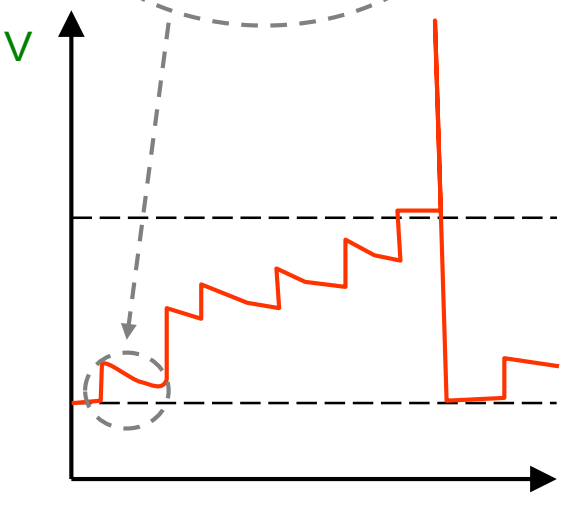


'Leaky' integrate-&-fire

time constant (τ), due to passive current leakage

(τ = CR)





Leakage => takes longer to reach firing threshold & inputs must arrive closer together in time to summate.

$C = q/V$, $CV = q$, $C\,dV/dt = I$; $V = IR$, $I = V/R$;

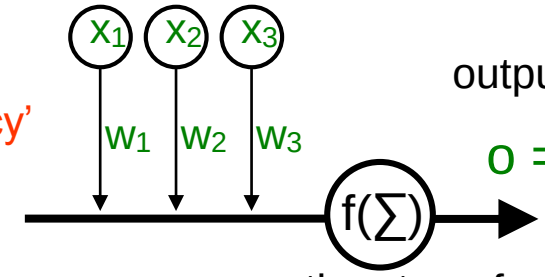
$C\,dV/dt = (V_r - V)/R + I$, with spike if $V = T$

Standard artificial neuron

Inputs (x_i): firing rates

Connection weights (w_i): net synaptic 'efficacy'

No physical structure, no simulation of spikes, connections can be +ve or -ve



output (o): firing rate

$o = f(\sum_i w_i x_i)$

The 'net input' to the neuron is:

$h = \sum_i w_i x_i = w_1x_1 + w_2x_2 + w_3x_3$,

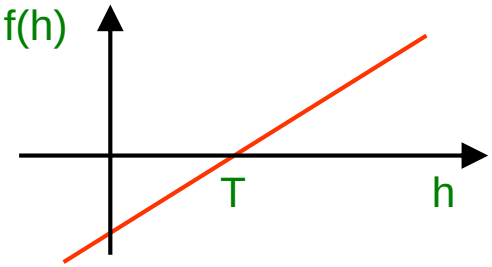
aka. the 'weighted sum' of input activation.

The 'transfer function' f(h) relates the output (o) to the net input: $o = f(h) = f(\sum_i w_i x_i)$ and includes the firing threshold T.

Common types of **transfer function**

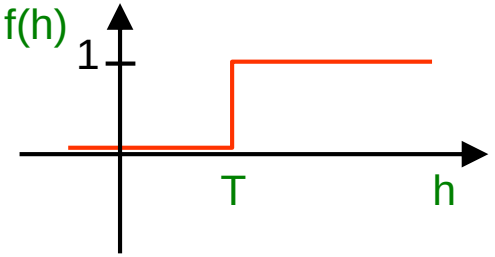
linear, e.g.

$f(h) = h-T$



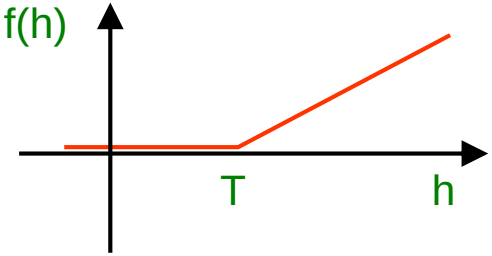
threshold logic function

$f(h)=\begin{cases} 1 & \text{if } h>T \\ 0 & \text{if } h<T \end{cases}$



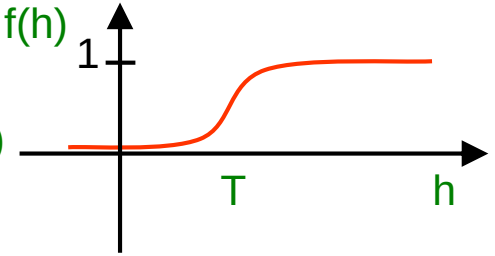
threshold-linear, e.g.

$f(h)=\begin{cases} h-T & \text{if } h>T \\ 0 & \text{if } h<T \end{cases}$



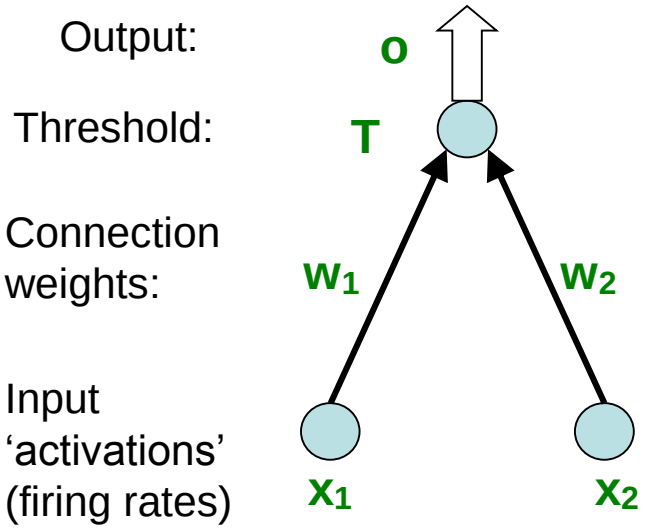
sigmoidal

$f(h)= \dfrac{1}{1+\exp(-(h-T))}$

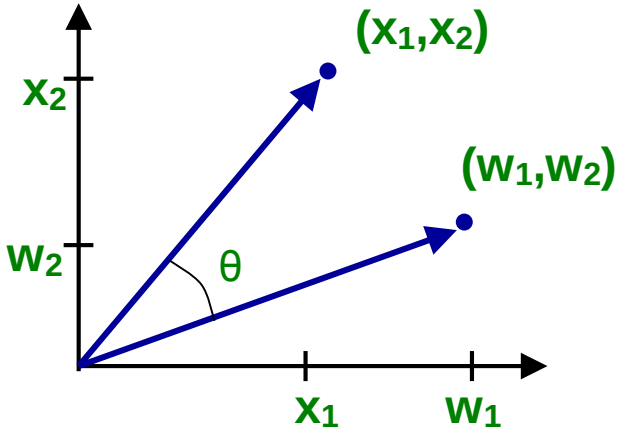


Implications of using the 'weighted sum' of input activations as the 'net input' to the artificial neuron

$o = f(h),\ h = w_1x_1 + w_2x_2$

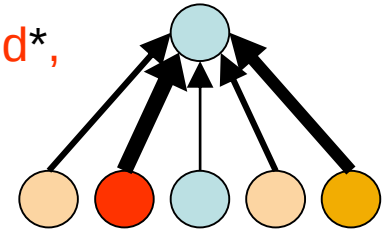


The input activations and the connection weights can be thought of as vectors $\underline{x}$ and $\underline{w}$.
The weighted sum $w_1x_1 + w_2x_2$ is also known as the 'dot product' ($\underline{w}.\underline{x}$) of $\underline{x}$ and $\underline{w}$ and depends on the angle θ between them: $\underline{w}.\underline{x} = |\underline{w}|\ |\underline{x}|\ \cos(\theta)$



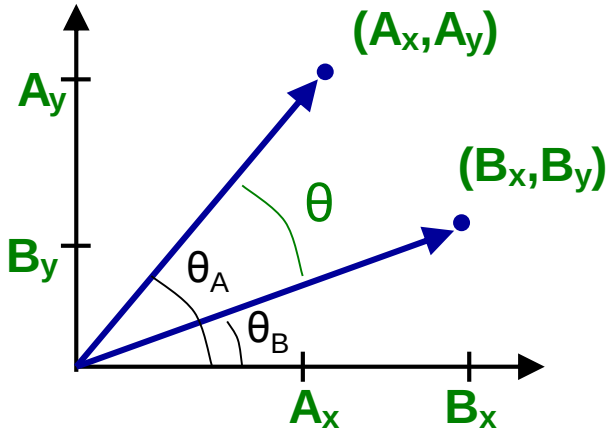
What this means is that..

If the total amounts of input activation & connection weight are limited*, the maximum net input h (& thus output firing rate) occurs when the patterns of input activations and of connection weights **match**.



* e.g. $|\underline{w}| = 1$, $|\underline{x}| = 1$.

The vector ‘dot product’



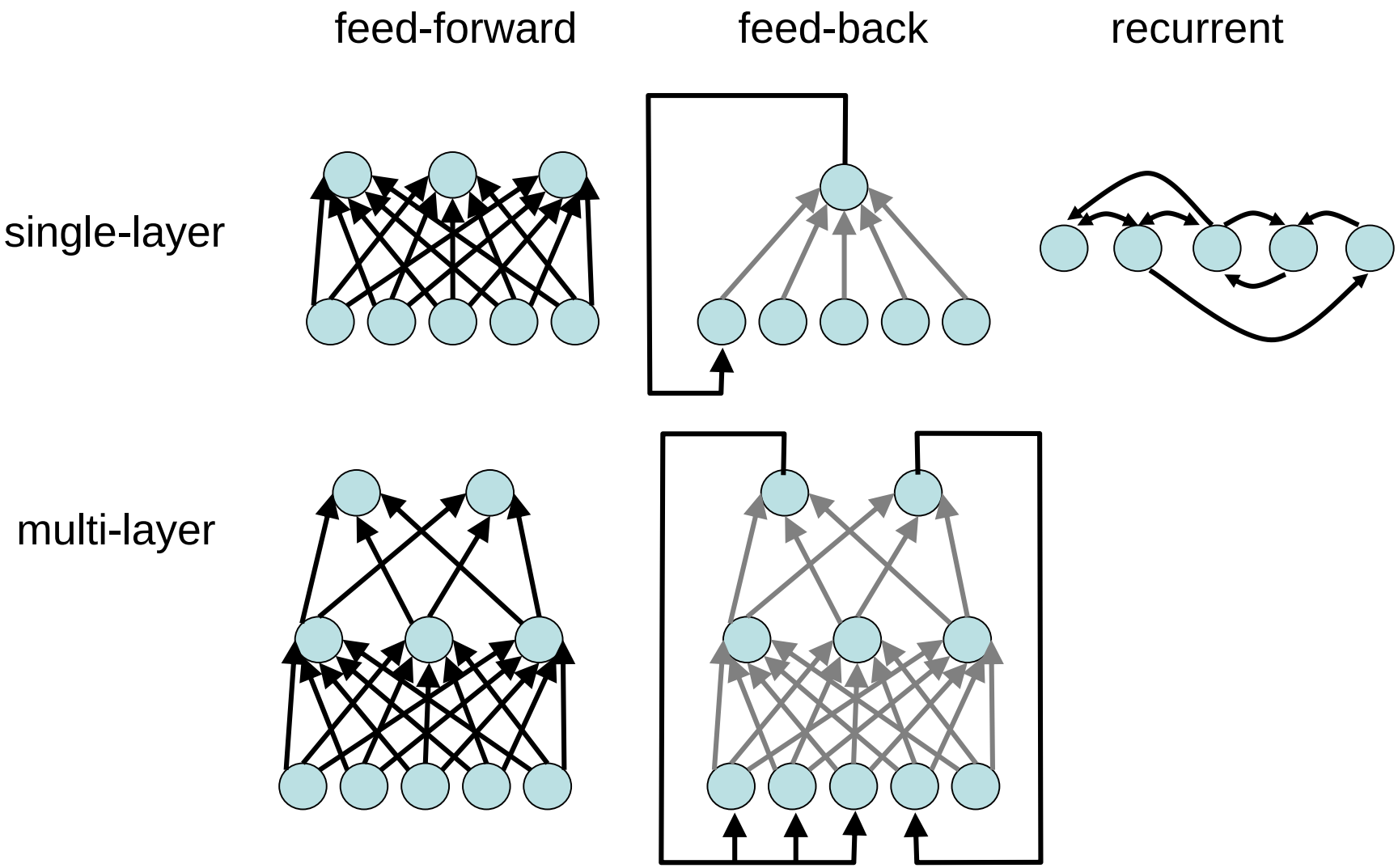
Definition: $\underline{A}.\underline{B} = |\underline{A}|\,|\underline{B}|\cos(\theta)$

So:
$$\underline{A}.\underline{B} = |\underline{A}|\,|\underline{B}|\cos(\theta_A-\theta_B)$$
$$= |\underline{A}|\,|\underline{B}|\,(\cos(\theta_A)\cos(\theta_B) + \sin(\theta_A)\sin(\theta_B))$$
$$= |\underline{A}|\cos(\theta_A)\,|\underline{B}|\cos(\theta_B) + |\underline{A}|\sin(\theta_A)\,|\underline{B}|\sin(\theta_B)$$
$$= A_x B_x + A_y B_y$$

$A_x = |\underline{A}|\cos(\theta_A)$, $A_y = |\underline{A}|\sin(\theta_A)$
$B_x = |\underline{B}|\cos(\theta_B)$, $B_y = |\underline{B}|\sin(\theta_B)$

More generally: $\underline{A}.\underline{B} = \sum_i A_i B_i$

Types of artificial neural networks



Learning

The problem: find connection weights such that the network does something useful.

Solution:

Experience-dependent learning rules to modify connection weights, i.e. learn from examples.

1. 'Unsupervised' (no 'teacher' or feedback about right and wrong outputs)
2. 'Supervised':
   A. Evolution/genetic algorithms
   B. Occasional reward or punishment ('reinforcement learning')
   C. Fully-supervised: each example includes correct output.

Unsupervised learning

- The ‘Hebb rule’, often interpreted as: strengthen connections between neurons that tend to be active at the same time. (cf Hebb, 1949)

$$w_{ij} \longrightarrow w_{ij} + \Delta w_{ij}$$

e.g. $w_{ij} \longrightarrow w_{ij} + \varepsilon\, x_j\, x_i$

Δw_{ij}	x_i = 0	x_i = 1
x_j = 0	?	-
x_j = 1	-	+

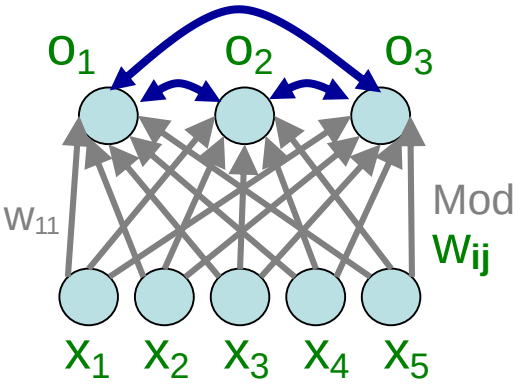
Cf. Long-term potentiation, long-term depression.

N.B. ANNs just model firing rate, so cannot implement more complex ‘spike-time dependent’ synaptic plasticity *(Bi &Poo, J Neurosci., 1998)*

Unsupervised learning, example 1:

Competitive learning

Fixed -ve connection weights



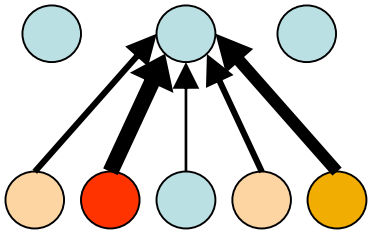
'Lateral inhibition' => one 'winner' if strongly activated enough
Rumelhart and Zipser (1986). 'Winner-takes-all' architecture.

Threshold linear units.

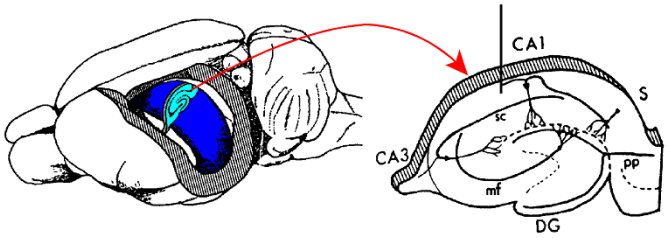
- Random initial connection weights
- Present n^{th} input pattern $\underline{x}^n$
- winner: output o_k (i.e. $h_k > h_i$ for all $i \neq k$)
  set $o_k = 1$, $o_i = 0$ for all $i \neq k$
- Hebbian learning: $w_{ij} \rightarrow w_{ij} + \varepsilon\, o_i\, x_j^n$
  i.e. $w_{kj} \rightarrow w_{kj} + \varepsilon x_j^n$, other weights don't change.
- Normalisation: reduce total size of connection weights to each output (so $|\underline{w}_i| = 1$) by dividing each by $|\underline{w}_i|$ or using alternative combined learning rule:
  $w_{kj} \rightarrow w_{kj} + \varepsilon(x_j^n - w_{kj})$
- Present next input pattern..

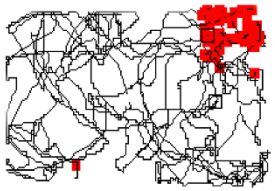
The output whose weights are most similar to $\underline{x}^n$ wins and its weights then become more similar. Different outputs find their own clusters in input data.

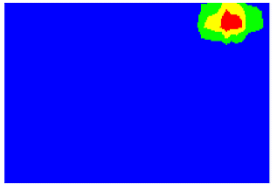
The output whose weights w are most similar to x^n wins, and its weights then become more similar. Different outputs find their own clusters in input data.



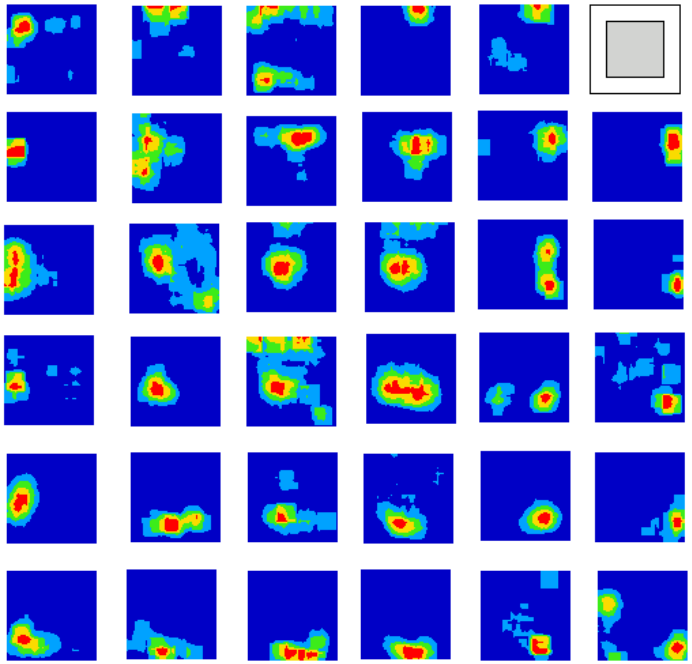
Sharp's (1991) model of place cell firing

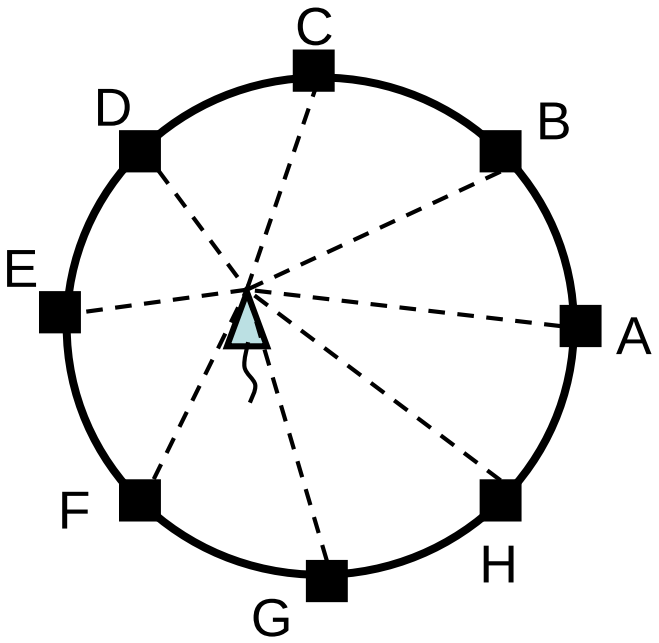




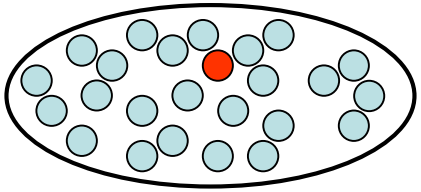


35 SIMULTANEOUSLY RECORDED PLACE CELLS





Competitive learning



Inputs:

far

left

near

right

A B C D E F G H

A B C D E F G H

Cue distances, directions

Competitive learning cont.

Competitive learning is built upon 3 ideas:

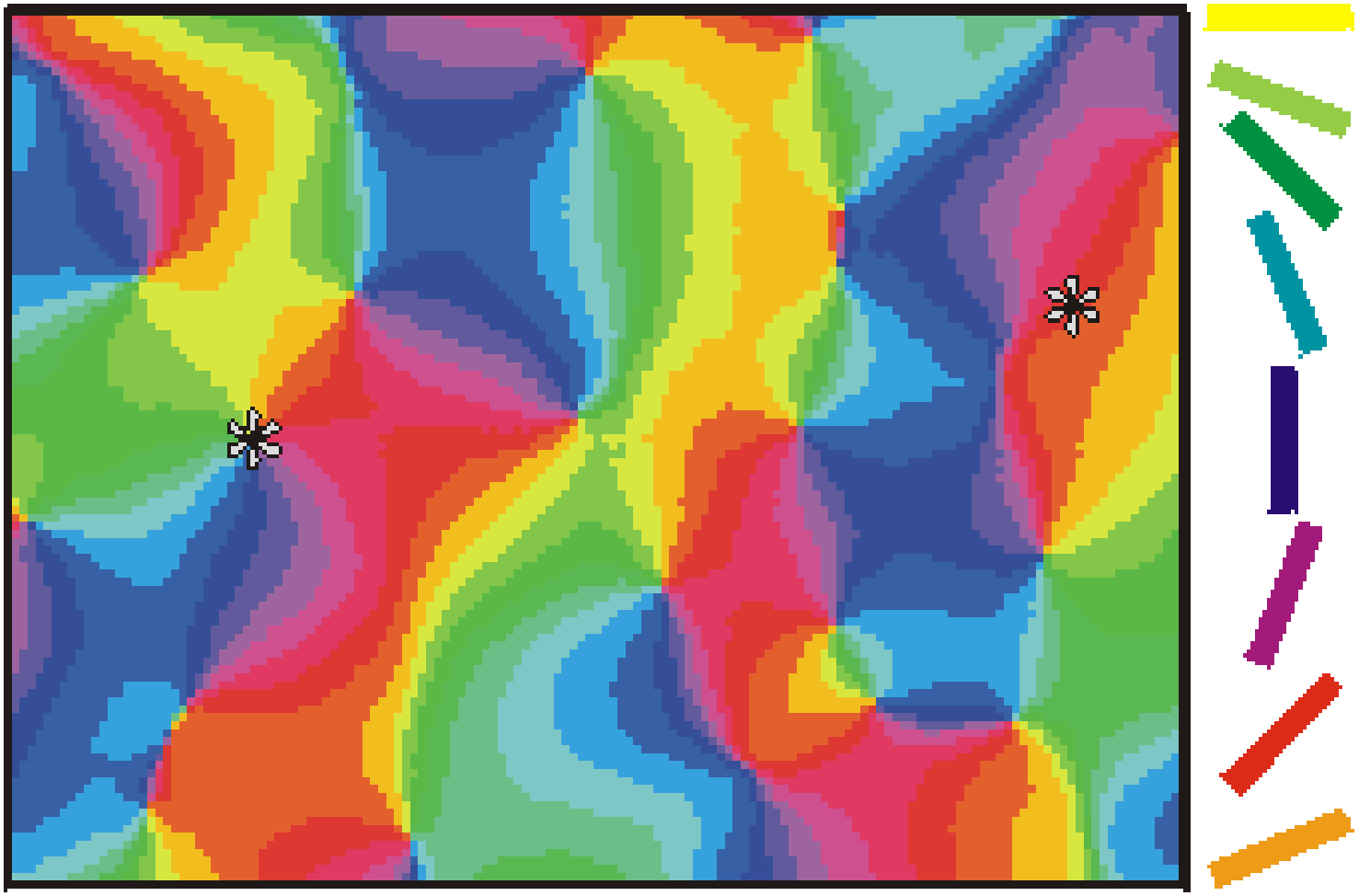
- **Hebbian learning principle:** when pre-synaptic and post-synaptic units are co-active, the connection between them should increase.
- **Competition between different units for activation**, through lateral inhibition / winner-take-all activation rule
- **Competition between incoming weights of a unit**, to prevent all weights from saturating, by *normalizing* the weights to have fixed net size: if some incoming weights to a unit grow, the others will shrink.

- Competitive learning performs **clustering** on the input patterns:
  - Each time a unit wins, it moves its weights closer to the current input pattern
  - A given unit will therefore be more likely to win the competition for similar inputs
  - Each unit's weights thereby move toward the centre of a cluster of input patterns

- See Chapter 5: "Feature Discovery by Competitive Learning", in Parallel distributed processing: explorations in the microstructure of cognition, edited by Rumelhart et al, 1986. Textbook pages 88-93.

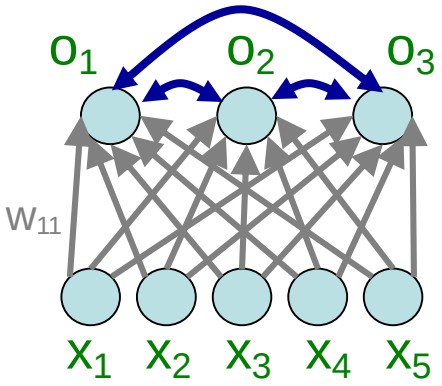
- Example of competitive learning: Sharp's model of place cell firing.

Topographic organisation of orientation selectivity in V1

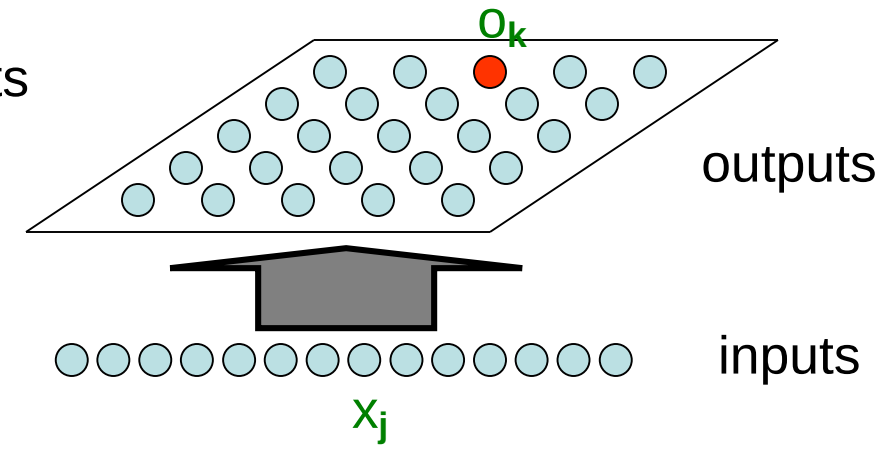


Unsupervised learning, example 2:

Feature maps & self-organisation

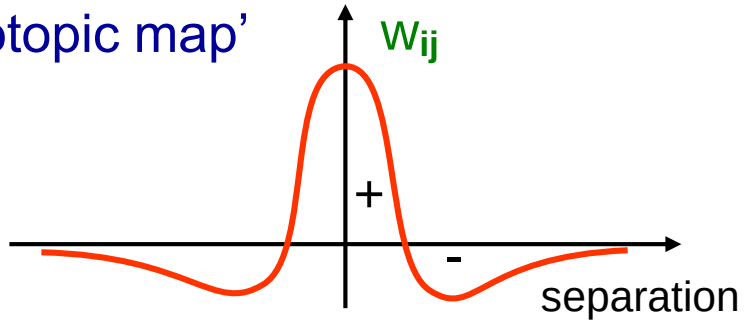


Arrange output units in a sheet:



Willshaw and Von der Marlsburg's (1976) 'retinotopic map'

Lateral connections (between outputs) vary with neurons' separation - excitatory nearby, inhibitory far apart ('mexican hat' function):



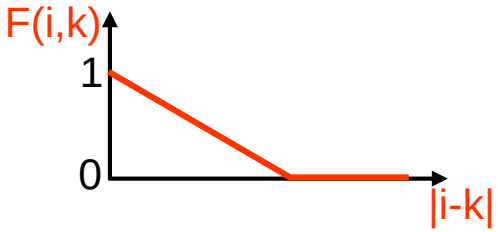
- Works like competitive learning, but not only 1 winner active: nearby units also active and so also learn to respond to similar input patterns.
- Produces a 2D map of the similarities present in a large set of input patterns.

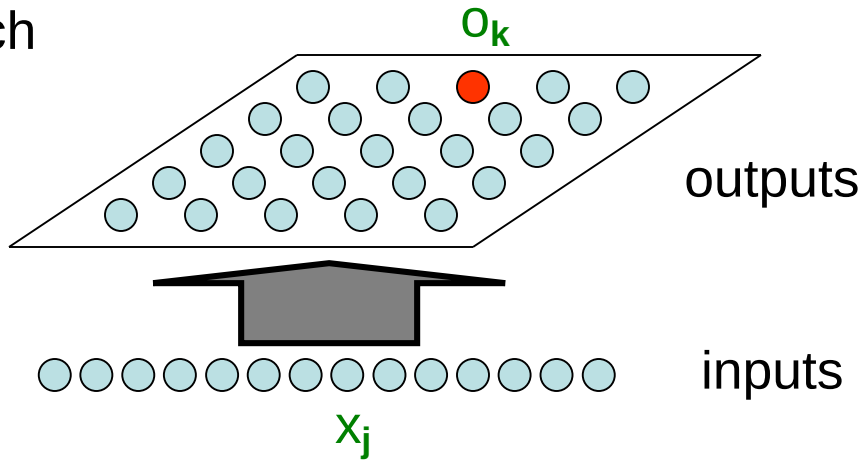
Kohonen's feature map (1982)

1 winner-takes-all as in competitive learning, but learning rule modified so that weights to outputs neighbouring the winner (o_k) are also modified using a 'neighbourhood function' F.

Present many input patterns, for each change weights according to:

$w_{ij} \rightarrow w_{ij} + \varepsilon F(i,k)(x_j^n - w_{ij})$



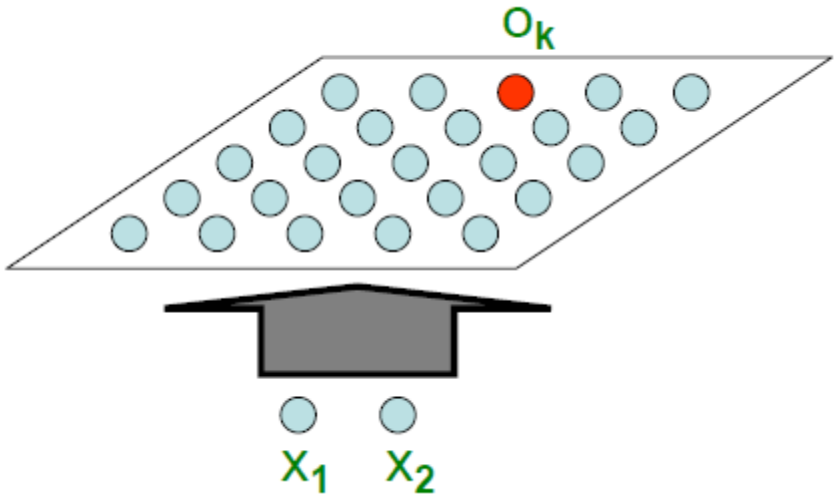


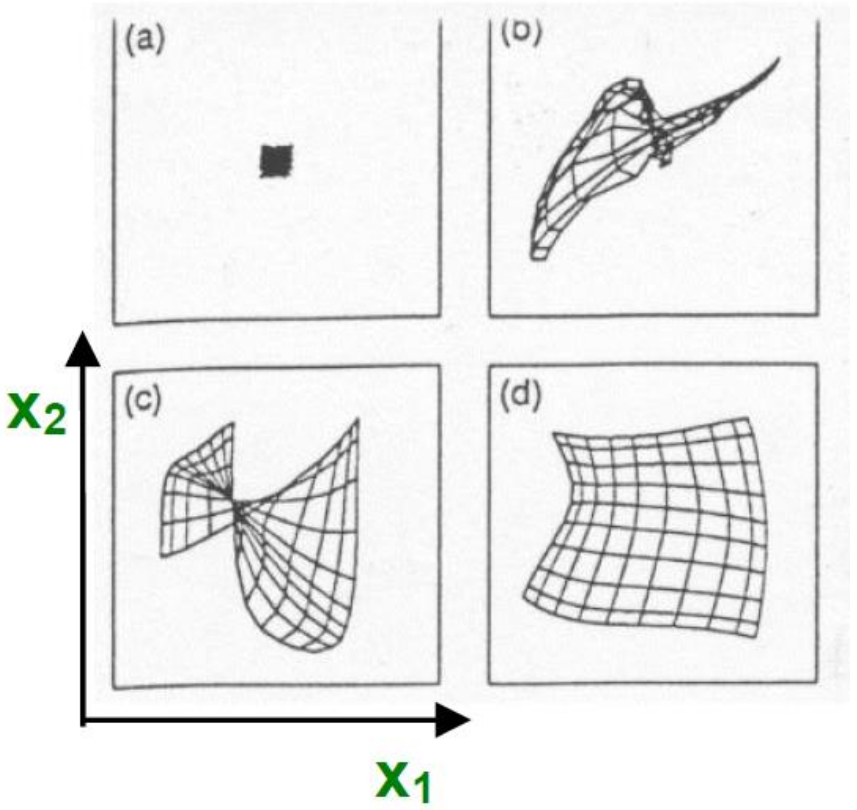
Causes nearby outputs to learn to represent (be active for) similar stimuli: producing a 2D map of complex (many D) data.

Cf. Learning in a volume, e.g. caused by the physical spread of chemical neurotransmitters or messengers, does not need (implausible?) lateral connections used by Willshaw & Von der Marlsburg.

Kohonen's feature map (1982), cont.

The structure of a map of 2-D data, and how it changes with learning, can be seen by showing each output unit in the part of input space that it represents:





Kohonen's feature map (1982), cont.

The structure of a map of high-dimensional data can be seen by labelling what each output unit represents:

The input for each animal is a long binary vector of its attributes (e.g. 2-feet, 4-feet, can swim, can fly, has feathers, eats meat etc etc).

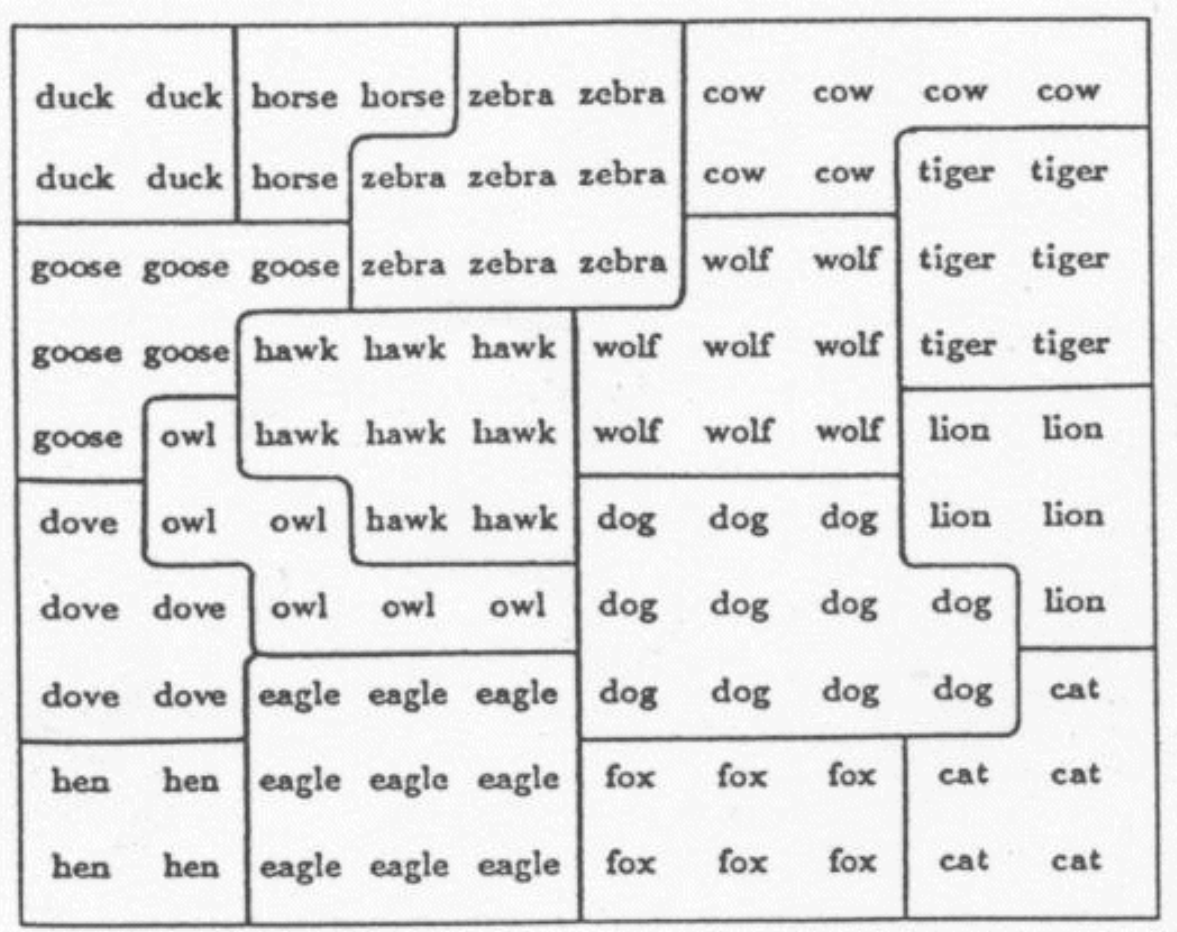


Figure 2. Visualization of a 10 × 10 feature map for a set of pattern vectors describing binary features of 16 animal species. The spatial arrangement of the labeled map regions reflects the similarity relationships between the animals.

SUMMARY: Introduction to Artificial Neural Networks, Unsupervised learning

1. An artificial neuron (v. simple model of a real neuron, McCulloch and Pitts, Rosenblatt..).

   Input values: x_i, connection weights: w_i, 'weighted sum' of inputs $\sum_i w_i x_i$, threshold T, output o; 'transfer function' $f(input)$.

2. Learning: How to find a useful set of connections w_{ij}:

   The Hebb rule and LTP: connection weight w_{ij} between neurons with activation x_i and x_j changes as $w_{ij} \rightarrow w_{ij} + \varepsilon\, x_i\, x_j$.

3. Unsupervised learning/ self-organisation in 'feed-forward' neural networks (NNs). Training set of input activations $\underline{x^k}$; each causes output activations o^k, and connection weights between active units are strengthened.

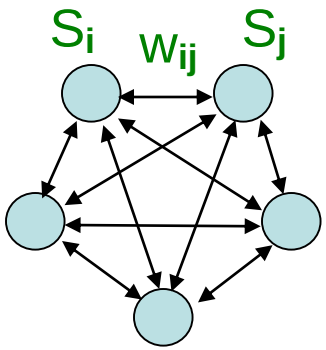
   (a) Competitive Learning (Rumelhart and Zipser, 1986). Lateral inhibition/ winner-take-all dynamics. Weight normalisation. Feature extraction: data clustering; Sharp's (1991) model of place field formation.

   (b) Feature Maps. 'Mexican hat' lateral connections, Willshaw and Von der Marlsburg's retinotopic map. Kohnonen's 'feature map': learning in a local volume (cf chemical diffusion?).

Unsupervised learning, example 3:

Hopfield's (1982) associative memory network

- Fully connected recurrent network (no input
- Symmetric connection weights ($w_{ij} = w_{ji}$)
- Units are active ($S_i = 1$) or inactive ($S_i = -1$)

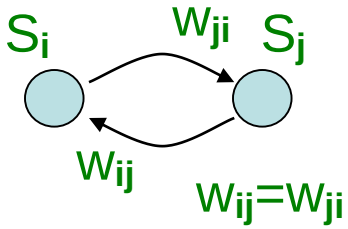


Learning: impose pattern of activation, use 'Hebbian' rule to change weights

$$w_{ij} \rightarrow w_{ij} + \varepsilon S_i S_j$$

Recall: start from similar pattern of activation, change activation according to sign of input to recover original pattern

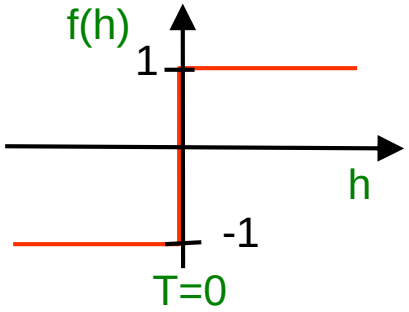
$$s_i = \text{sign}(\sum_j w_{ij} s_j)$$



Activation:

$s_i = f(h_i)$, where

$h_i = \sum_j w_{ij} s_j$

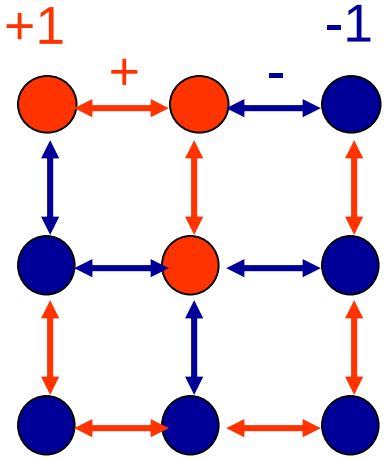


Learning:

Δw_{ij}	s_i = -1	s_i = 1
s_j = -1	↑	↓
s_j = 1	↓	↑

Hopfield networks, cont.

Patterns of activation are learned as ‘stable states’ under the rule for updating activations, e.g.



Δw_{ij}	s_i = -1	s_i = +1
s_j = -1	↑	↓
s_j = +1	↓	↑

Several different patterns can be learned in the same network, but the memory capacity is limited to about 0.14N.

Memory is ‘content addressable’: performing ‘pattern completion’ of partial cue. Spurious memories (combinations of real ones) are also formed.

More plausible learning rules show similar behaviour

Hopfield networks: attractors & stable states

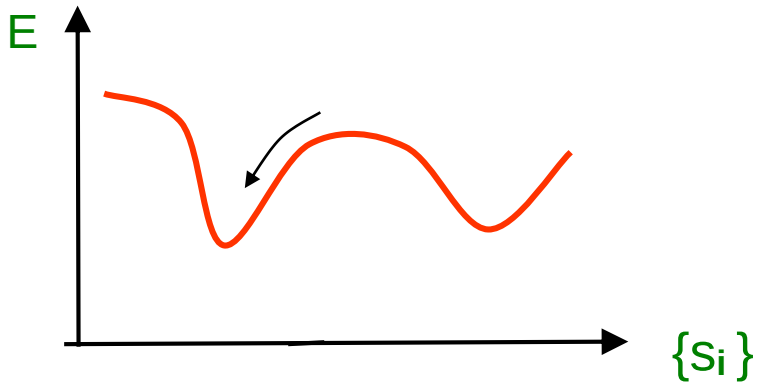
To support a pattern of activation connections should be positive between units in the same state (i.e. 11 or -1-1) and negative between units in different states (1-1 or -11), i.e. $s_i s_j w_{ij} > 0$

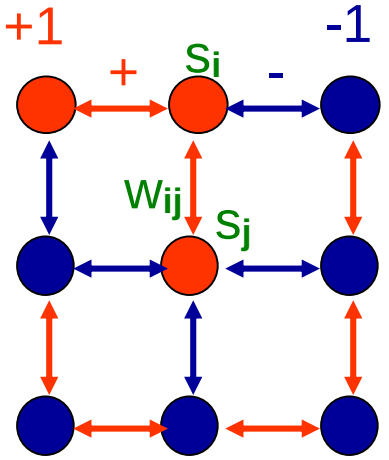
The 'frustration' or 'energy' of the system is how much this is not true, i.e.

$E = -\sum_{ij} s_i s_j w_{ij}$

The update rule changes each units activation to reduce the overall frustration, until the network ends up in a stable state from which it cannot be reduced further.

The learning rule sets the weights so that to-be-remembered patterns of activity are stable states (aka 'attractor states').

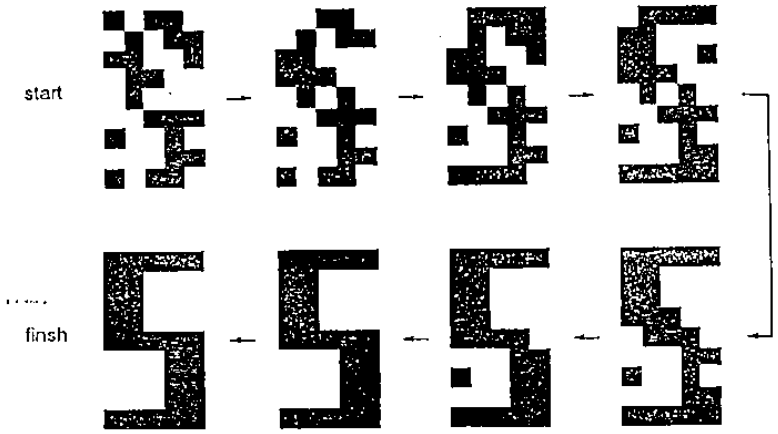




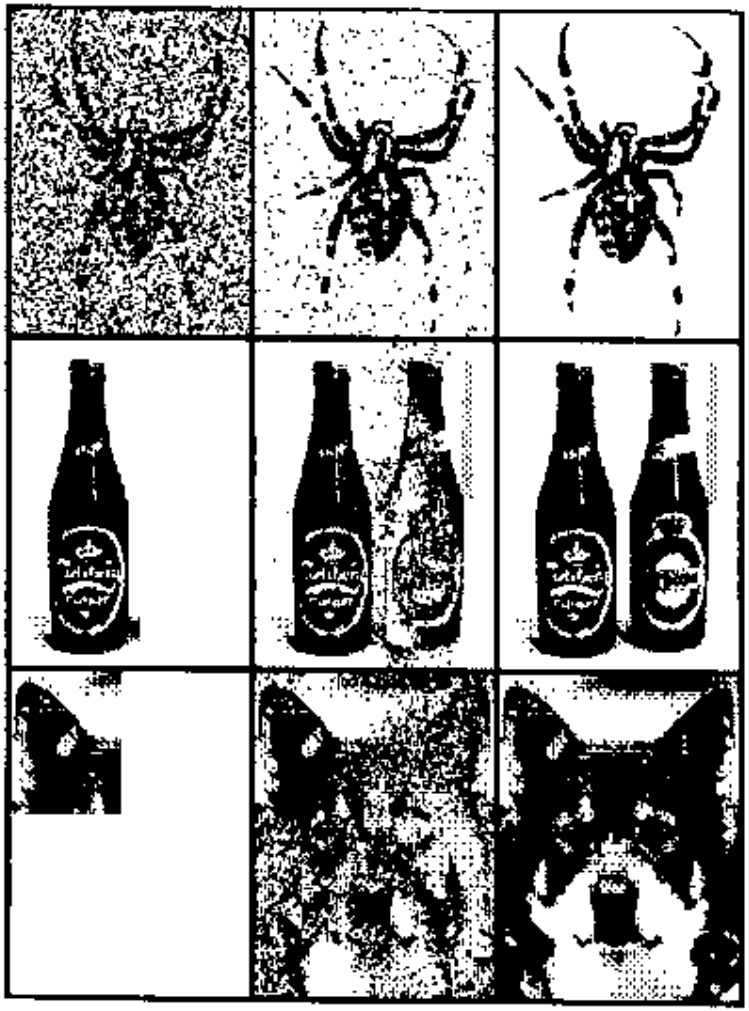
Examples of Hopfield networks

A 5x9 network storing 8 patterns

Figure 6.3 The training set for the Hopfield network. BeJ, 143



Retrieval in a 130x180 network



Hopfield networks, cont.

- **Activation rule**:
  - If net input is greater than zero, unit gets an activation of 1; otherwise activation is -1.
  - random, asynchronous update of activations
- **Architecure**:
  Symmetrically connected recurrent network.
- **Hebbian learning**:
  For each training pattern,
  - Set states of units to corresponding elements of pattern.
  - Increment each weight in proportion to product of pre- and post-synaptic states.
- **Desirable features:**
  - Attractor dynamics: guaranteed convergence to an attractor state.
  - Pattern completion
- **Undesirable features:**
  - Spurious attractors
  - Limited storage capacity