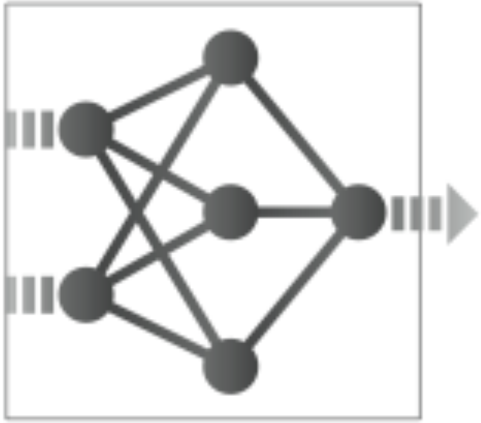
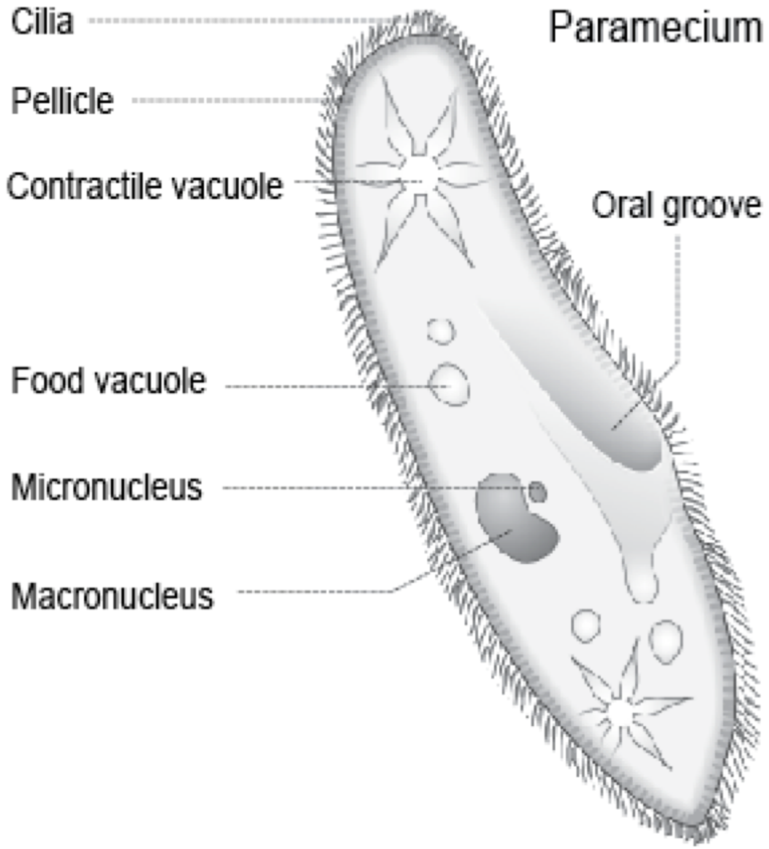
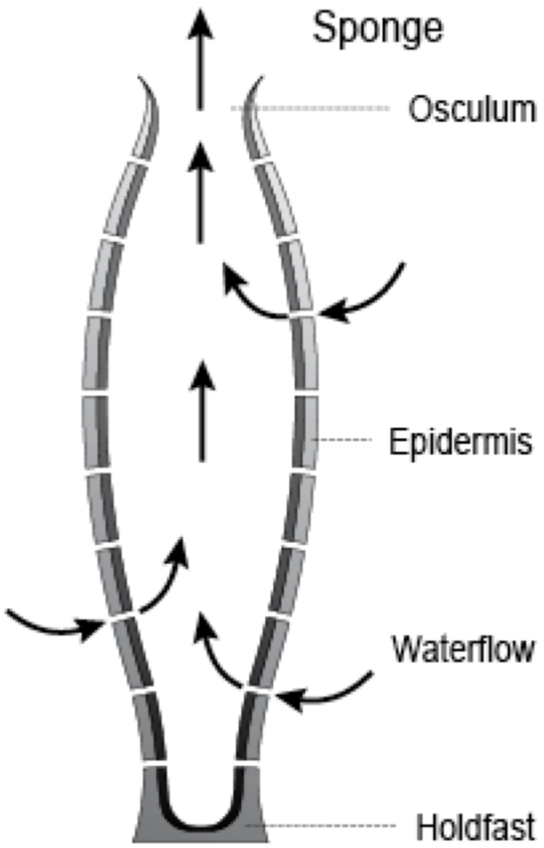
Neural Systems



Part 1

Do animals need nervous systems?

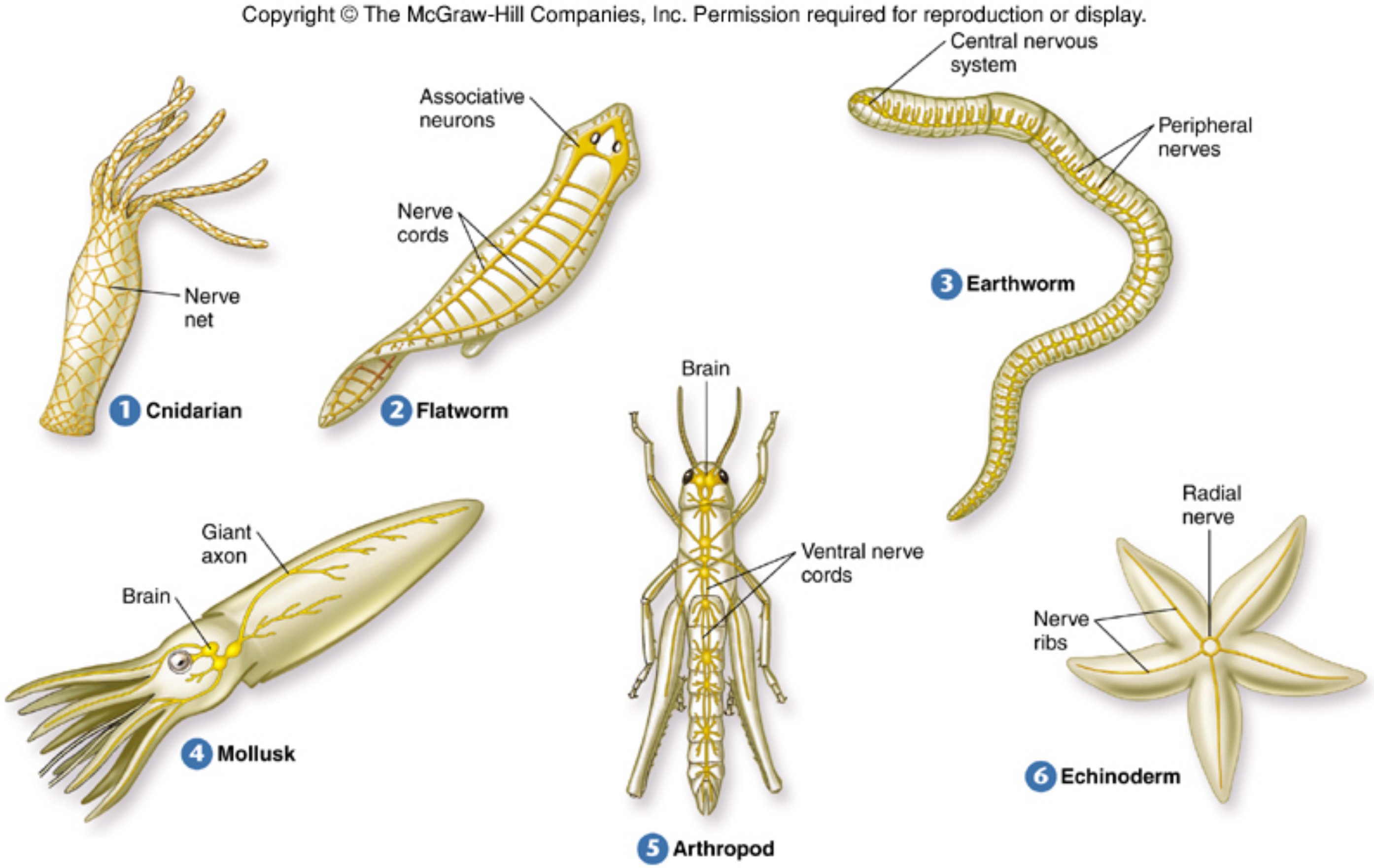




Not all animals have nervous systems; some use only chemical reactions

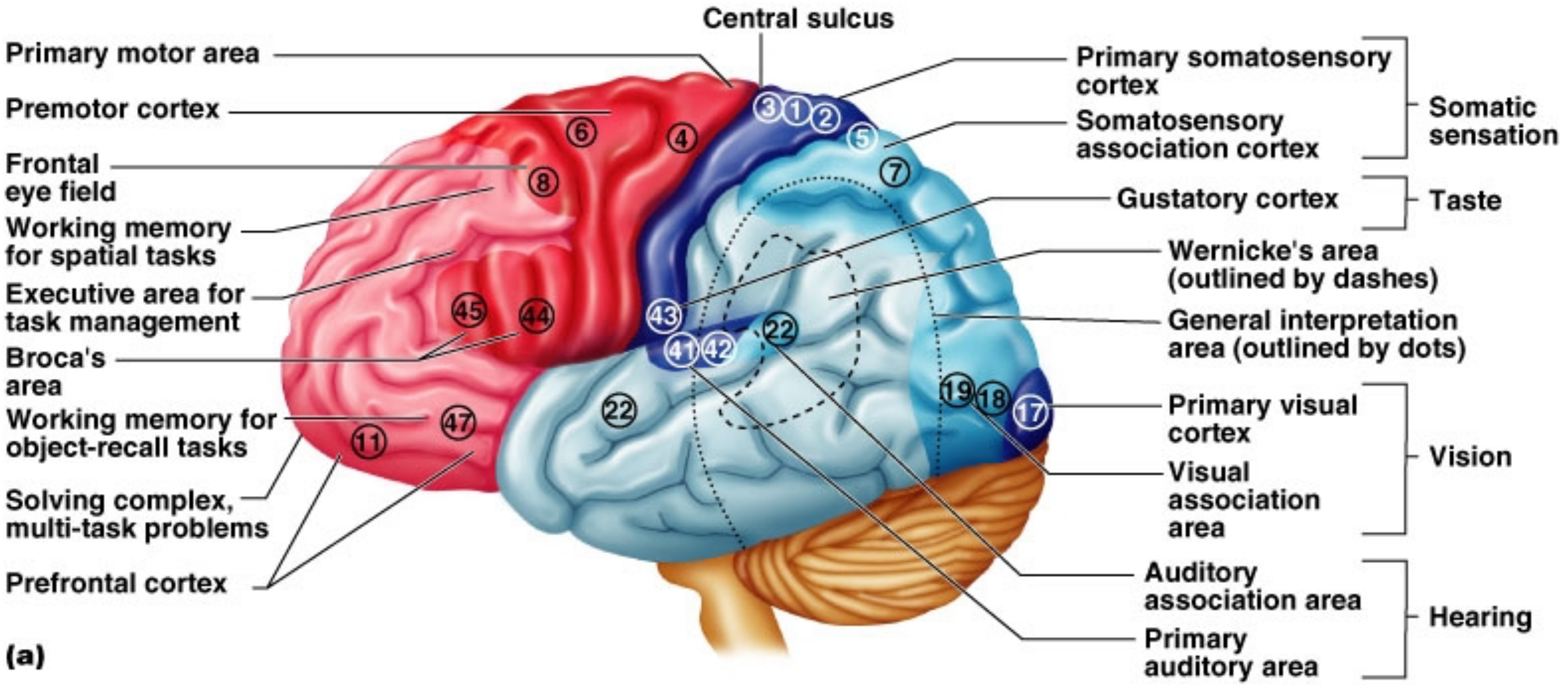
Paramecium and sponge move, eat, escape, display habituation

Why Nervous Systems?



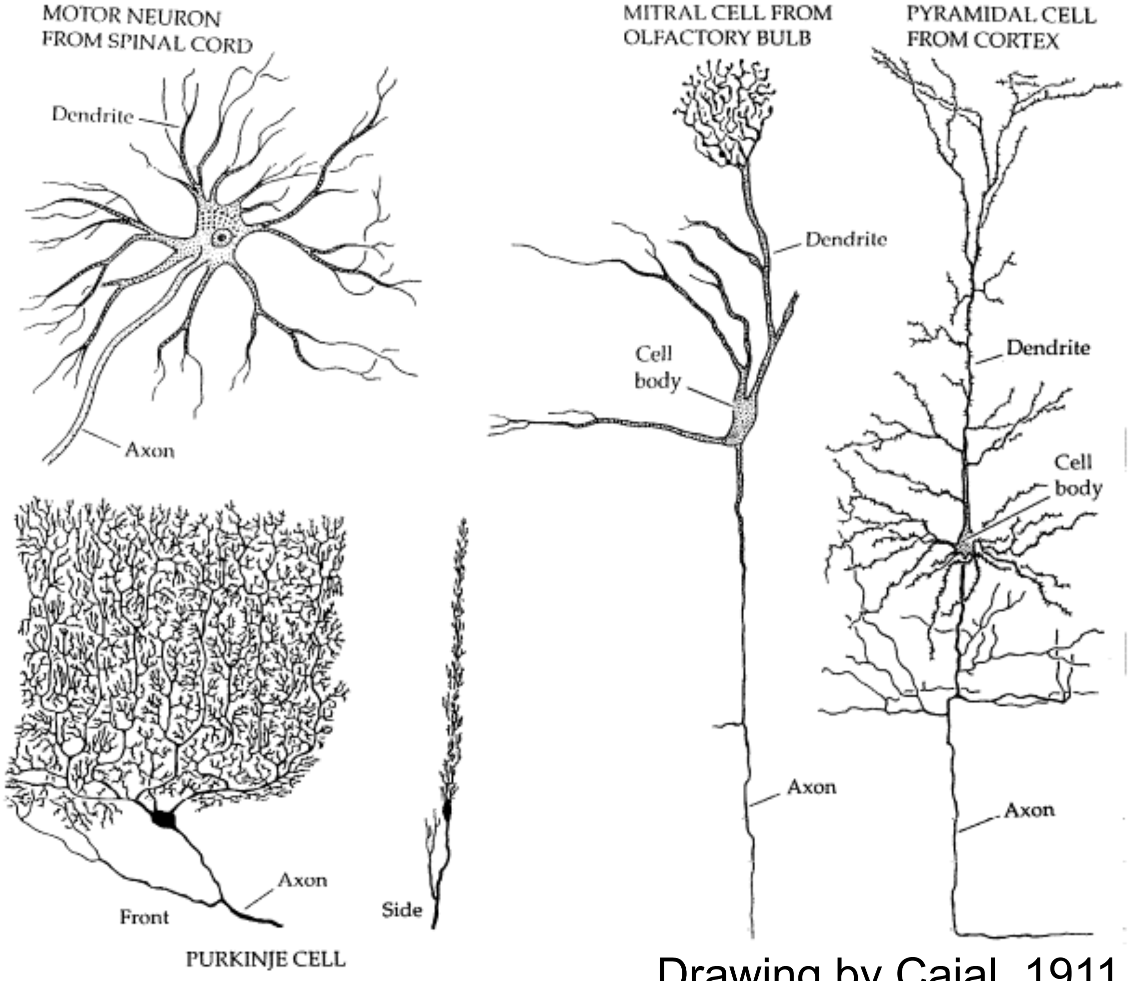
1) Faster reaction times = competitive advantage
2) Selective transmission of signals across distant areas = more complex bodies
3) Generation of non-reactive behaviors
4) Complex adaptation = survival in changing environments

Central Nervous System with Cortex


Central sulcus
Primary motor area
Premotor cortex
Frontal eye field
Working memory for spatial tasks
Executive area for task management
Broca's area
Working memory for object-recall tasks
Solving complex, multi-task problems
Prefrontal cortex
(a)
Primary somatosensory cortex
Somatosensory association cortex
Somatic sensation
Gustatory cortex
Taste
Wernicke's area (outlined by dashes)
General interpretation area (outlined by dots)
Primary visual cortex
Visual association area
Vision
Auditory association area
Primary auditory area
Hearing
3
1
2
5
7
6
4
8
45
44
43
41
42
22
22
47
11
19
18
17

What Does Make Brains Different?

Components and behavior of individual neurons are very similar across animal species and, presumably, over evolutionary history (Parker, 1919)

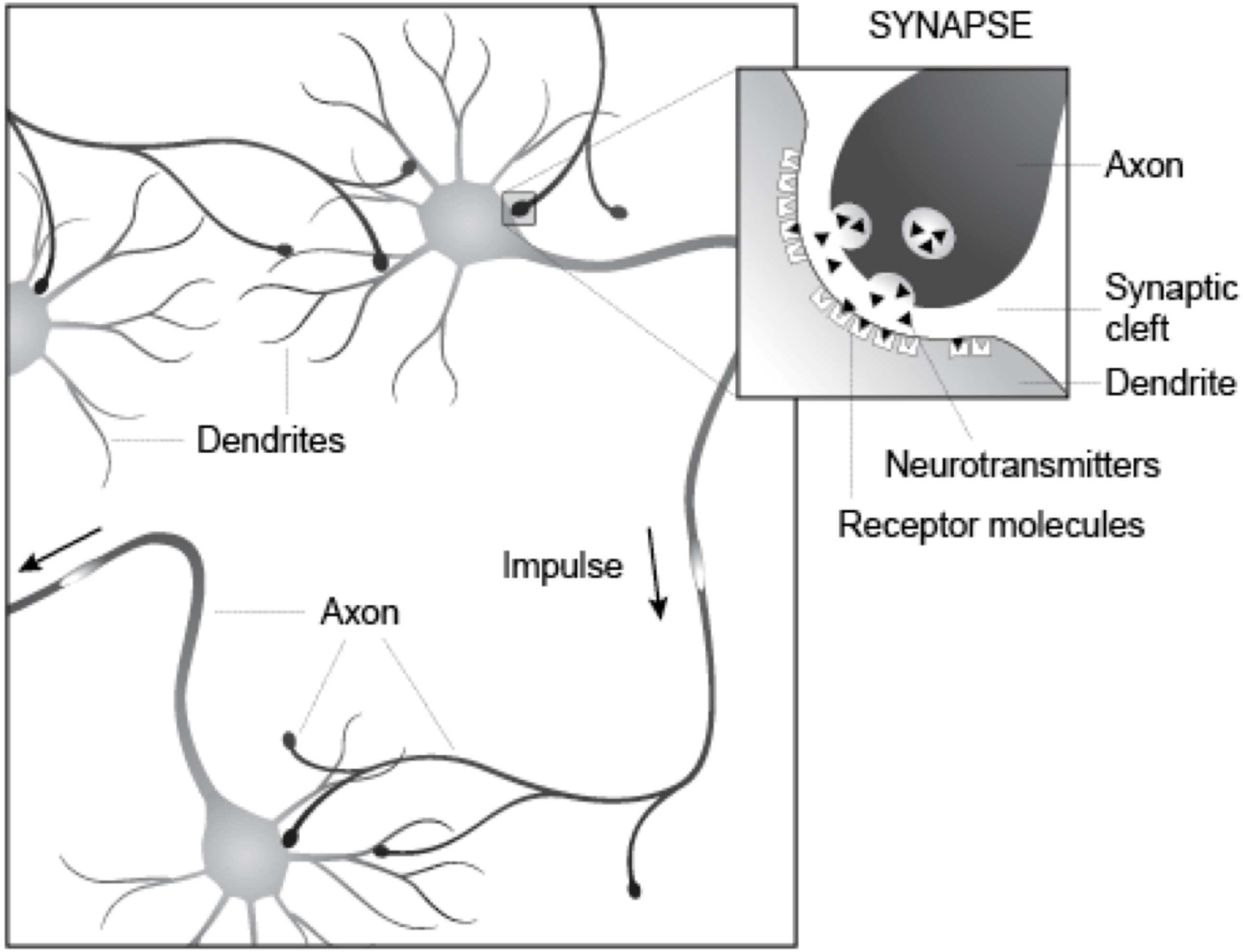

MOTOR NEURON FROM SPINAL CORD
Dendrite
Axon
MITRAL CELL FROM OLFACTORY BULB
Dendrite
Cell body
Axon
PYRAMIDAL CELL FROM CORTEX
Dendrite
Cell body
Axon
Axon
Front
Side
PURKINJE CELL

Drawing by Cajal, 1911

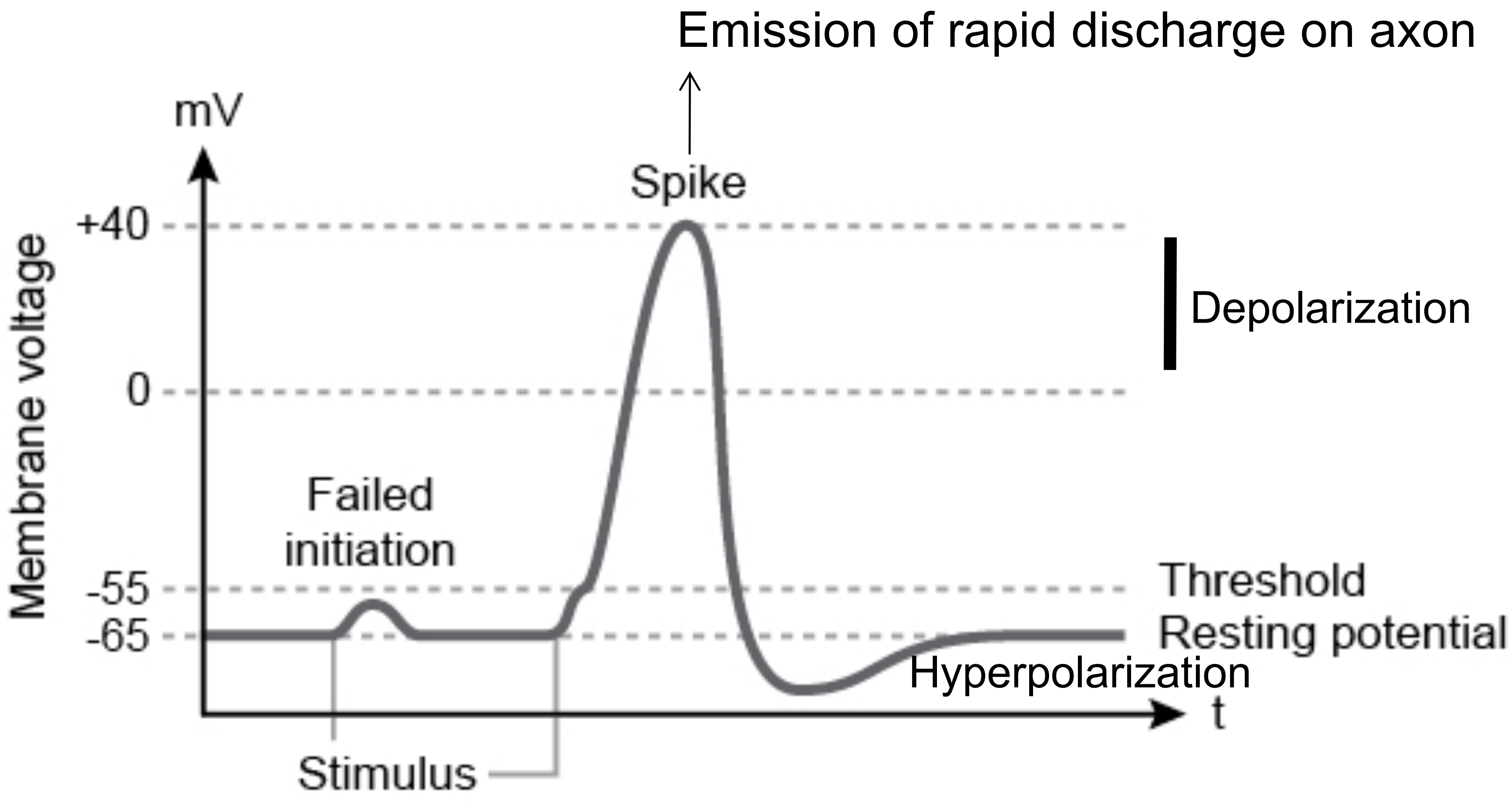
Evolution of the brain seems to occur mainly in the **architecture**, that is how neurons are interconnected.

First classification of neurons by Cajal in 1911 was made according to their connectivity patterns

Biological Neurons

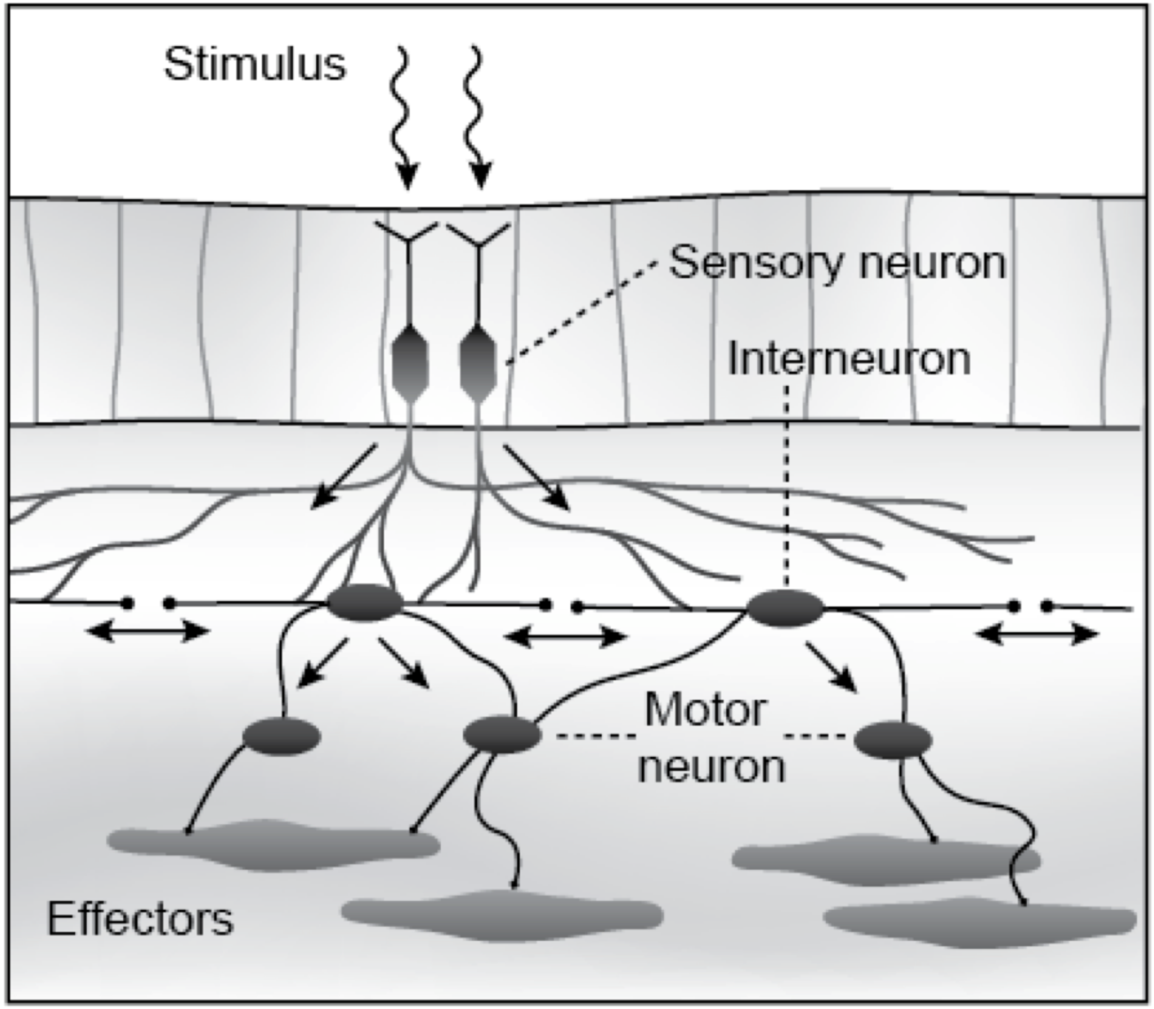


Membrane Dynamics



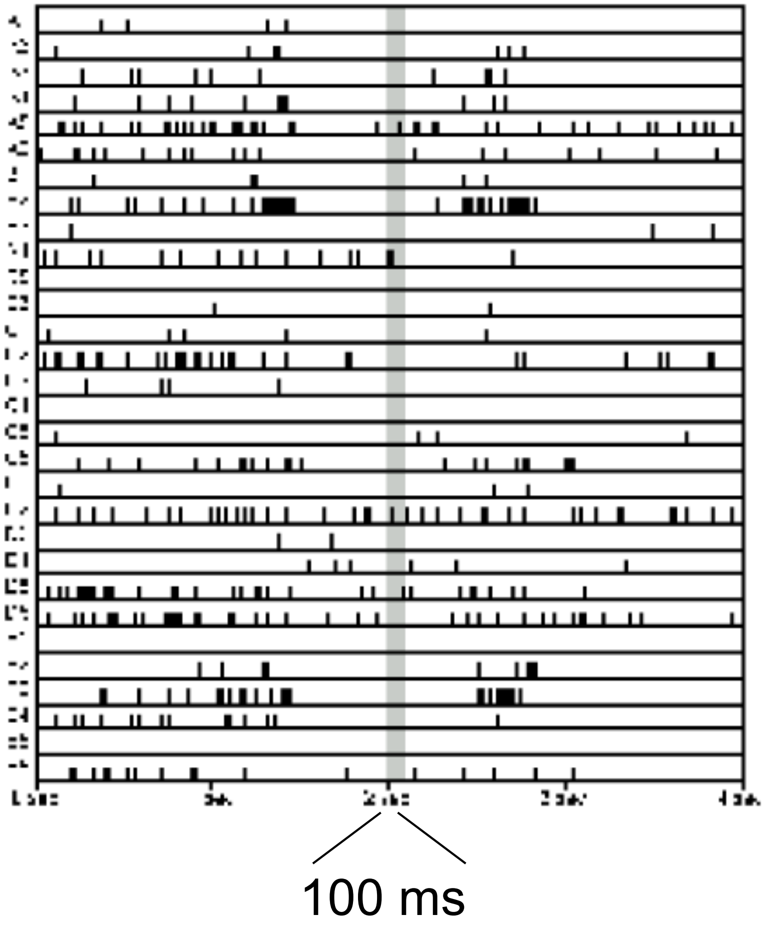
This cycle lasts approximately 3-50 ms, depending on type of ion channels involved (Hodgkin and Huxley, 1952)

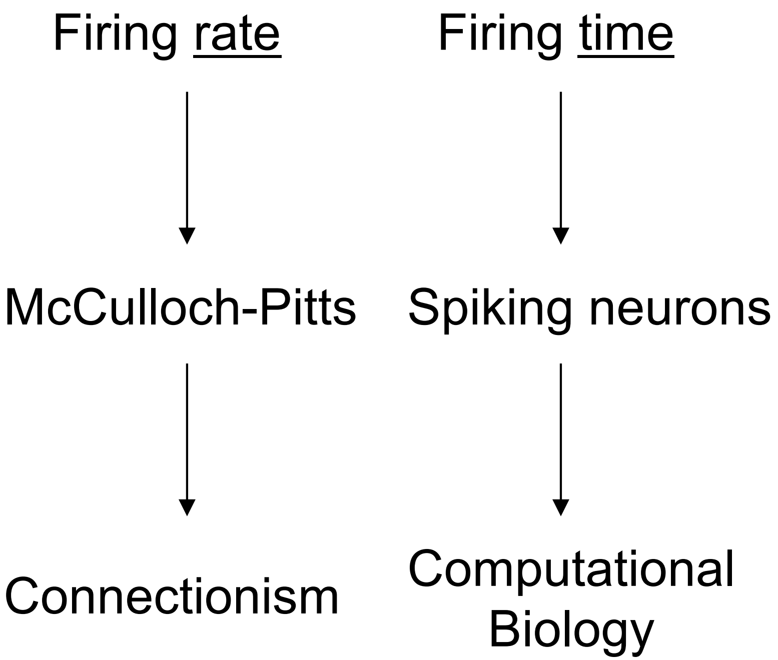
Types of Neurons



Interneurons can be
1- Excitatory
2- Inhibitory

How Do Neurons Communicate?



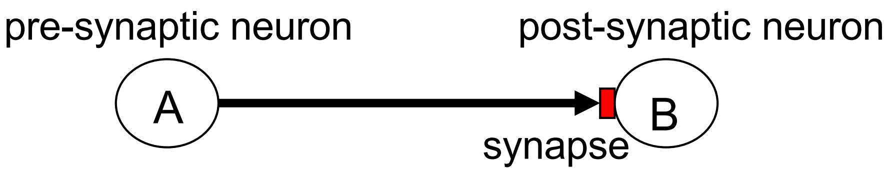


Firing rate → McCulloch-Pitts → Connectionism

Firing time → Spiking neurons → Computational Biology

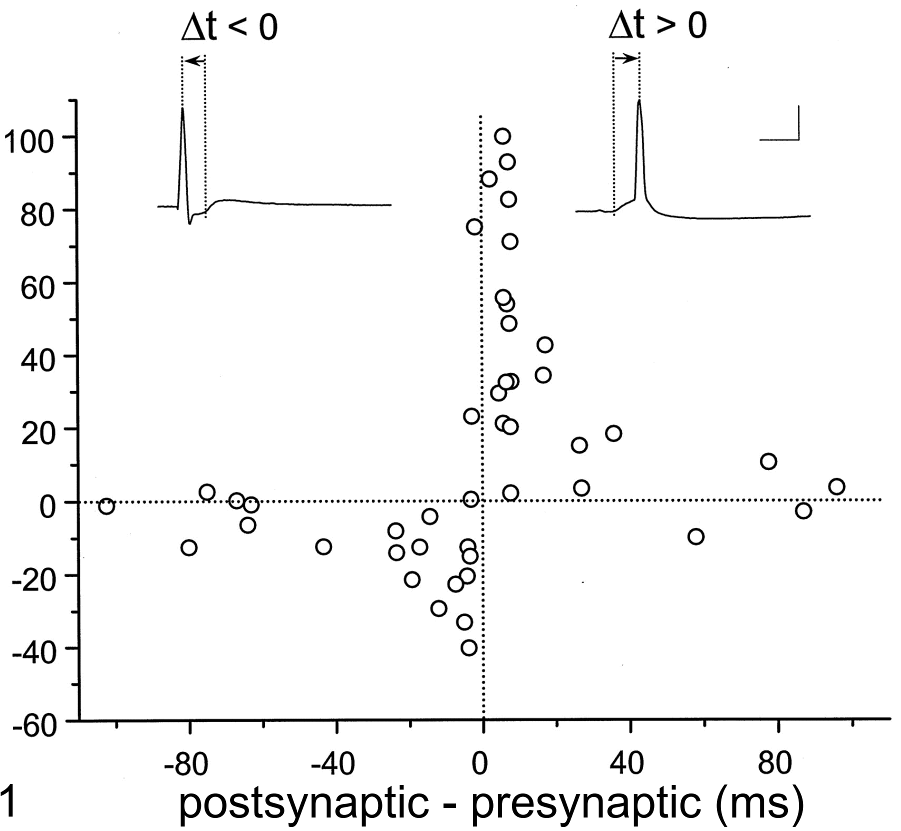
How Do Neurons Learn?

They learn by means of synaptic change



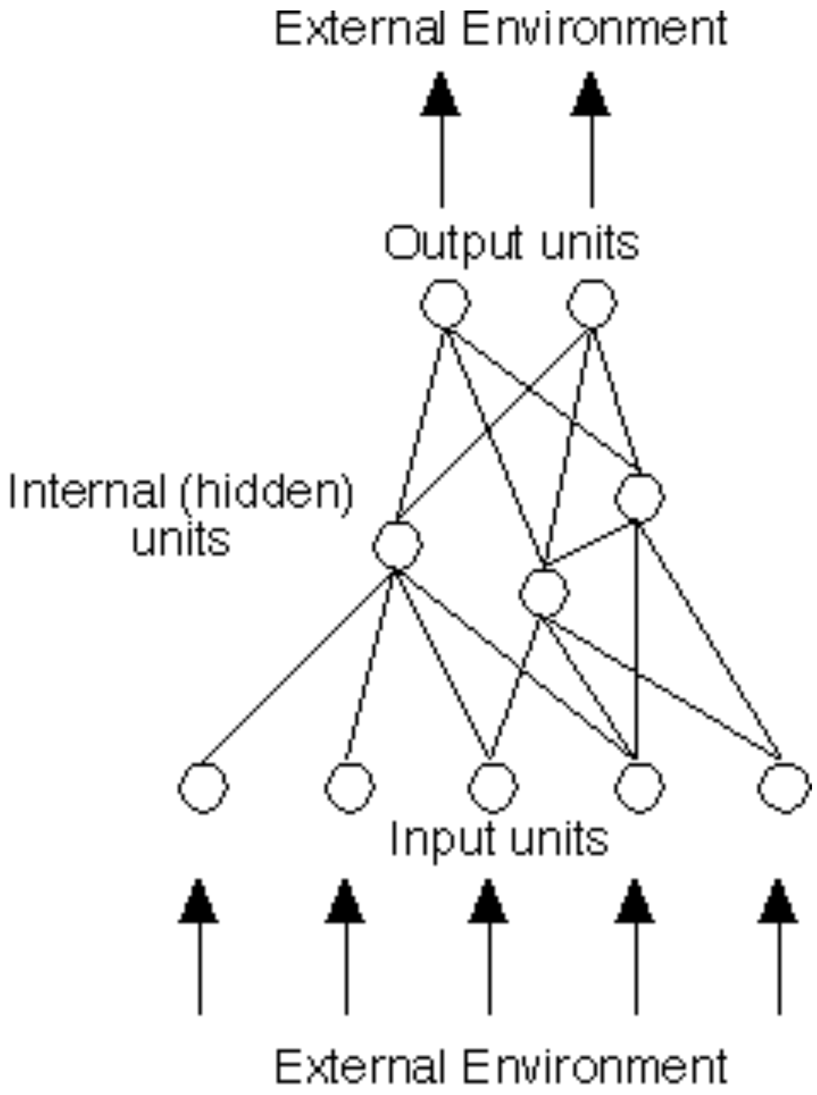
Hebb rule (1949):
Synaptic strength is increased if cell A consistently contributes to firing of cell B
This implies a temporal relation: neuron A fires first, neuron B fires second

Spike Time Dependent Plasticity (STDP):
- Small time window
- Strengthening (LTP) for positive time difference
-Weakening (LTD) for negative time difference



From Bi and Poo, 2001

An Artificial Neural Network

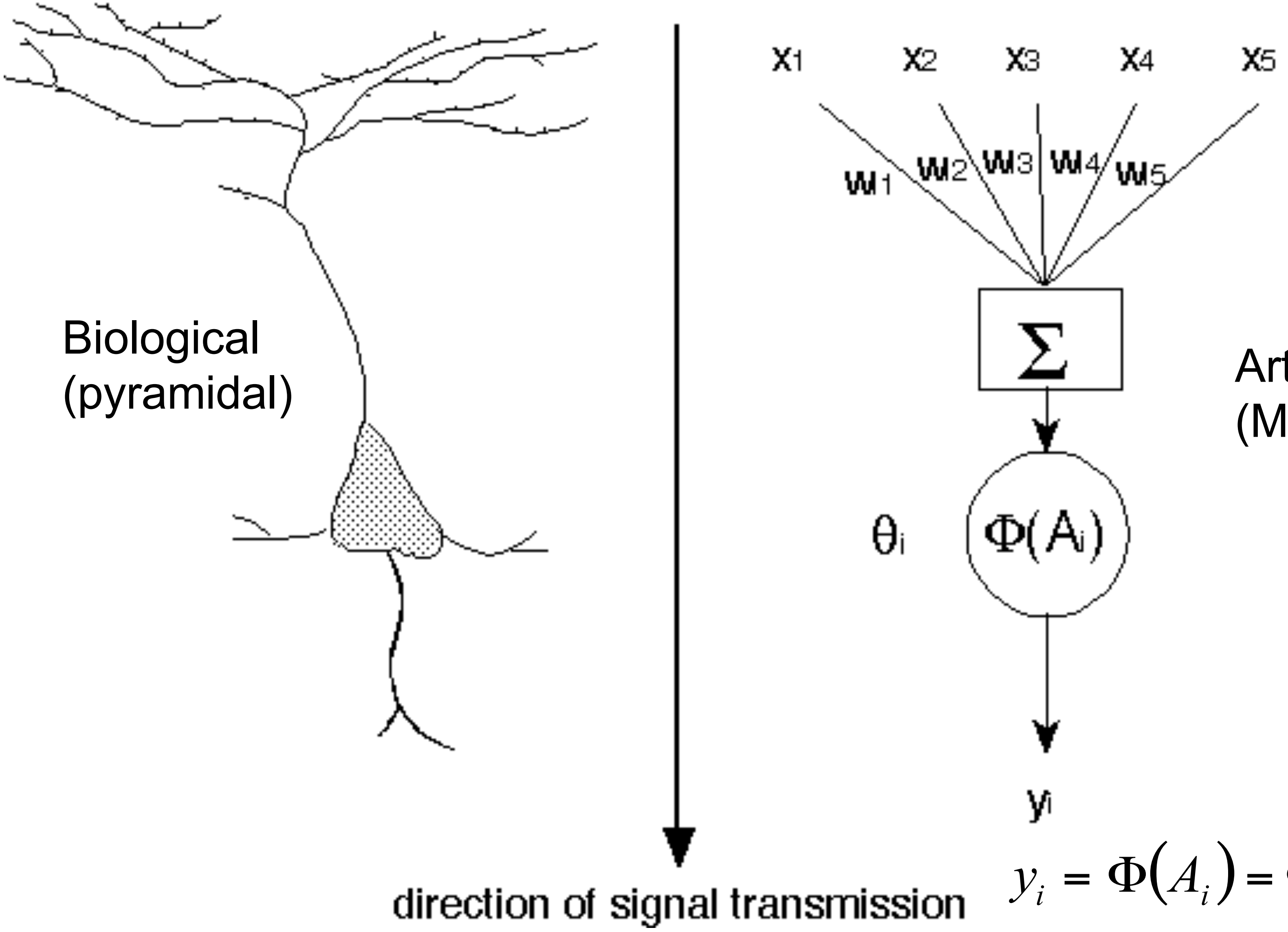


A neural network communicates with the environments through input units and output units. All other elements are called internal or hidden units.

Units are linked by uni-directional connections.

A connection is characterized by a weight and a sign that transforms the signal.

Biological and Artificial Neurons



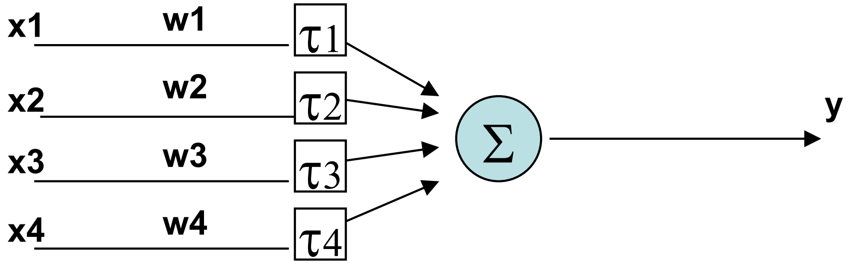
Biological
(pyramidal)

Artificial
(McCulloch-Pitts)

$$y_i = \Phi(A_i) = \Phi\left(\sum_{j}^{N} w_{ij} x_j - \vartheta_i\right)$$

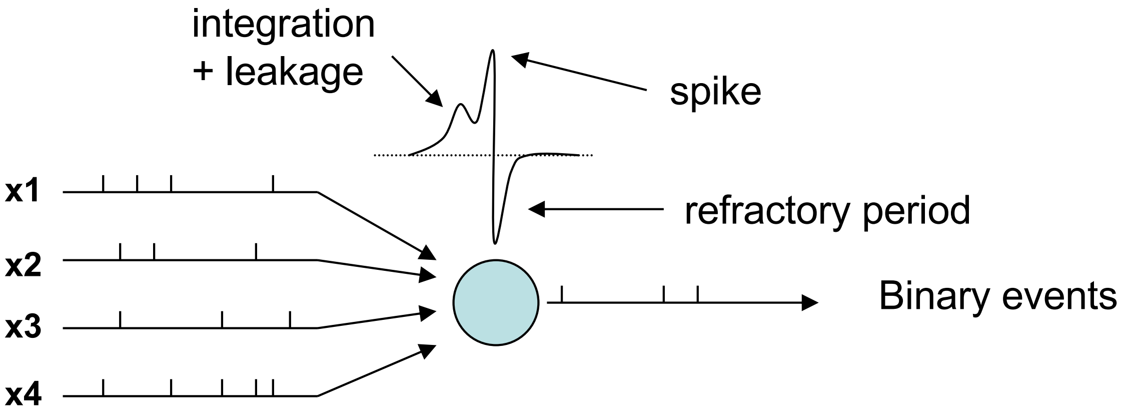
Neuron models

McCulloch-Pitts



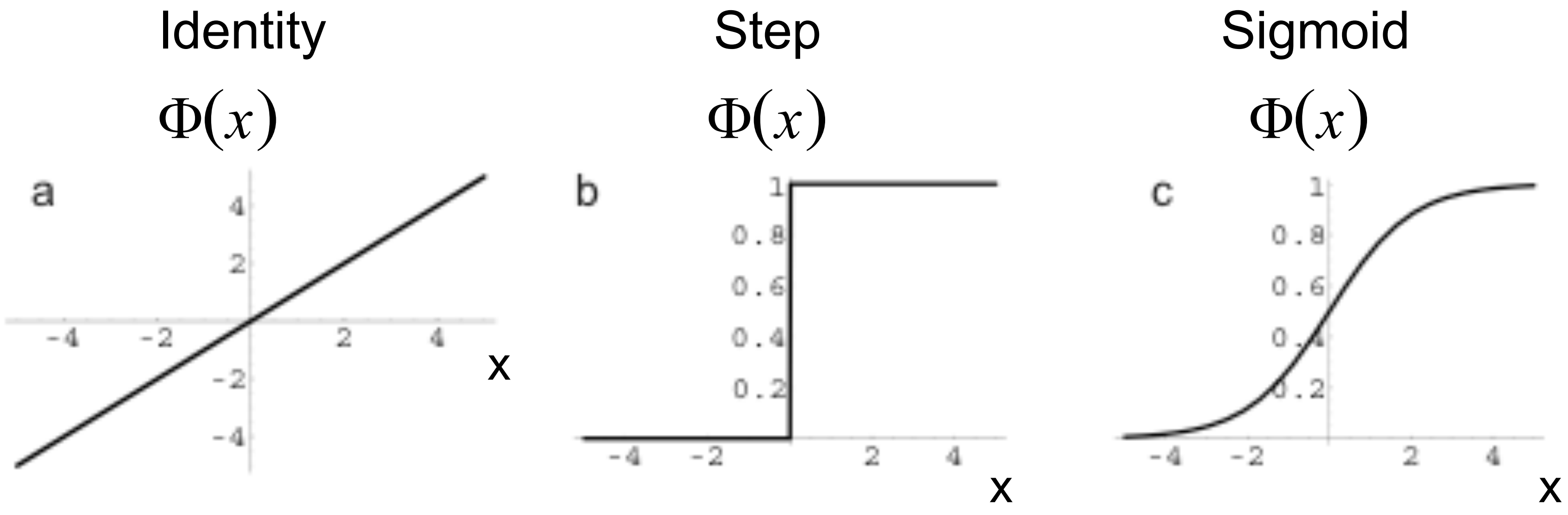
Continuous values

Spiking



Binary events

Output functions



Sigmoid function:

- continuous
- non-linear
- monotonic
- bounded
- asymptotic

$$\Phi(x) = \frac{1}{1+e^{-kx}}$$

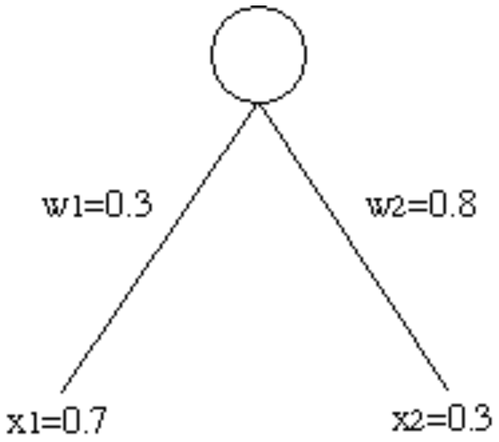
$$\Phi(x) = \tanh(kx)$$

Neurons signal “familiarity”

The output of a neuron is a measure of similarity between its input pattern and its pattern of connection weights.

1. Output of a neuron in linear algebra notation:

$$y = a\left(\sum_{i}^{N} w_i x_i\right), \qquad a = 1 \longrightarrow y = \mathbf{w} \cdot \mathbf{x}$$



2. Distance between two vectors is:

$$\cos\vartheta = \frac{\mathbf{w}\cdot\mathbf{x}}{\|\mathbf{w}\|\|\mathbf{x}\|}, \qquad 0 \le \vartheta \le \pi$$

where the vector length is:

$$\|\mathbf{x}\| = \sqrt{\mathbf{x}\cdot\mathbf{x}} = \sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}$$

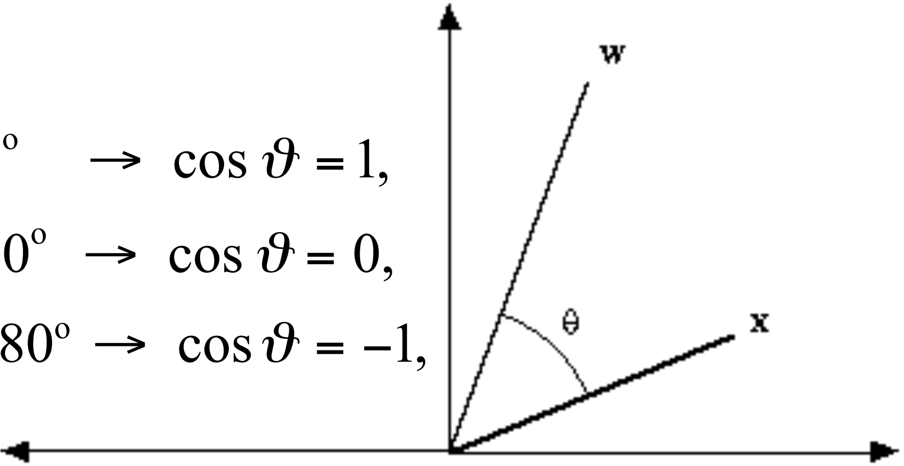
3. Output signals input familiarity

$$\mathbf{w}\cdot\mathbf{x} = \|\mathbf{w}\|\|\mathbf{x}\|\cos\vartheta$$

$$\vartheta = 0^{\circ} \rightarrow \cos\vartheta = 1,$$

$$\vartheta = 90^{\circ} \rightarrow \cos\vartheta = 0,$$

$$\vartheta = 180^{\circ} \rightarrow \cos\vartheta = -1,$$

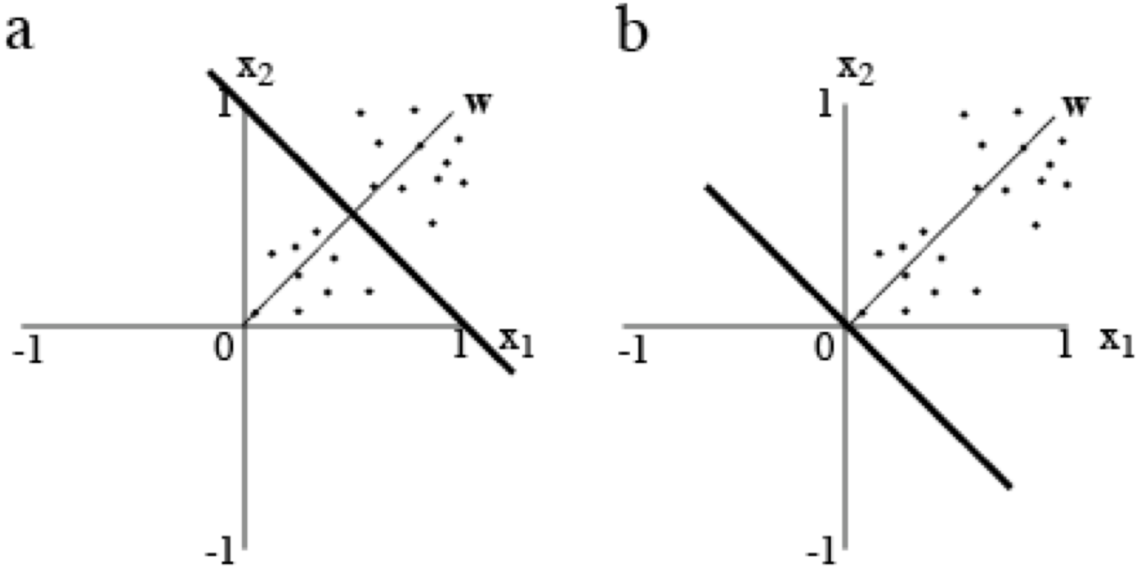


Neurons as classifiers

A binary neuron divides the input space in two regions, one where weighted input sum >=0 and one where weighted input sum <0.

The separation line is defined by the synaptic weights:

$$w_1 x_1 + w_2 x_2 - \vartheta = 0 \qquad x_2 = \frac{\vartheta}{w_2} - \frac{w_1}{w_2} x_1$$



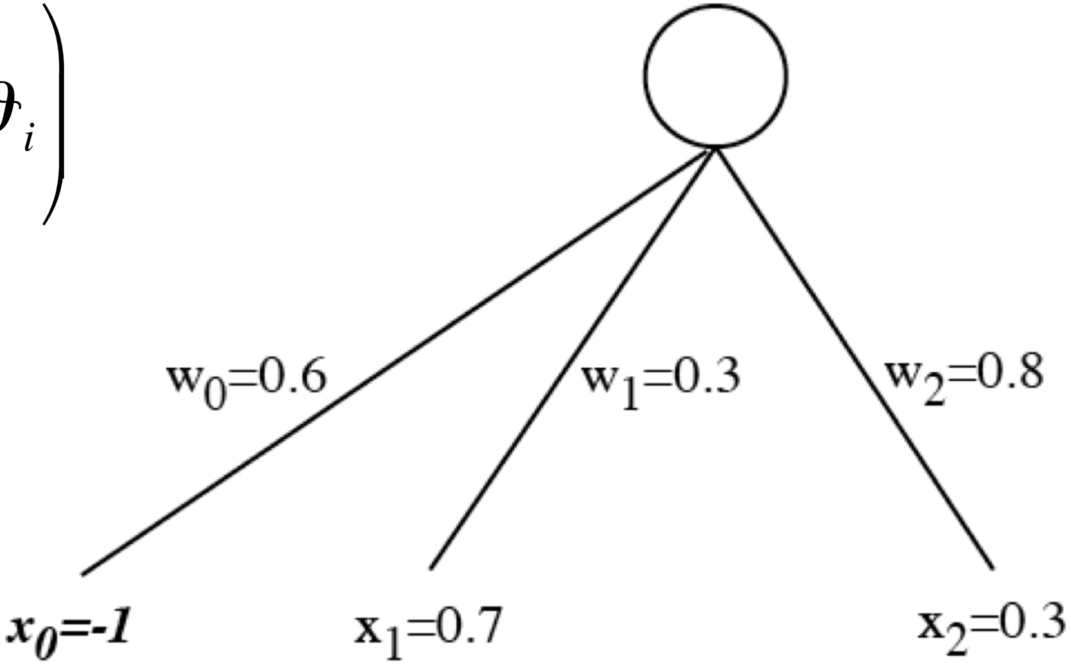
$\vartheta > 0$ $\qquad$ $\vartheta = 0$

From Threshold to Bias unit

The threshold can be expressed as an additional weighted input from a special unit, known as bias unit, whose output is always -1.

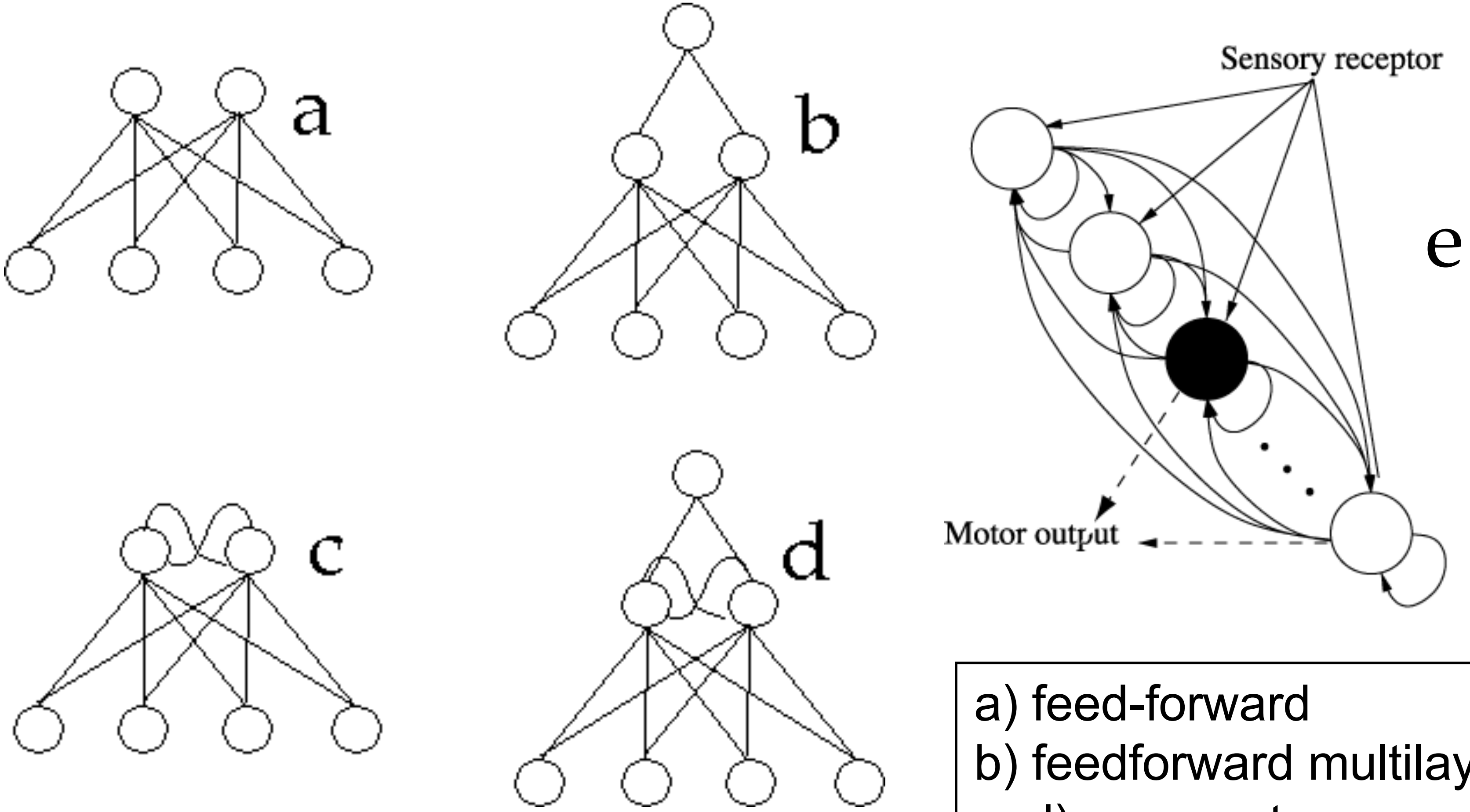
$$y_i = \Phi(A_i) = \Phi\left(\sum_{j=1}^{N} w_{ij} x_j - \vartheta_i\right)$$

$$y_i = \Phi(A_i) = \Phi\left(\sum_{j=0}^{N} w_{ij} x_j\right)$$



- Easier to express/program
- Threshold is adaptable like other weights

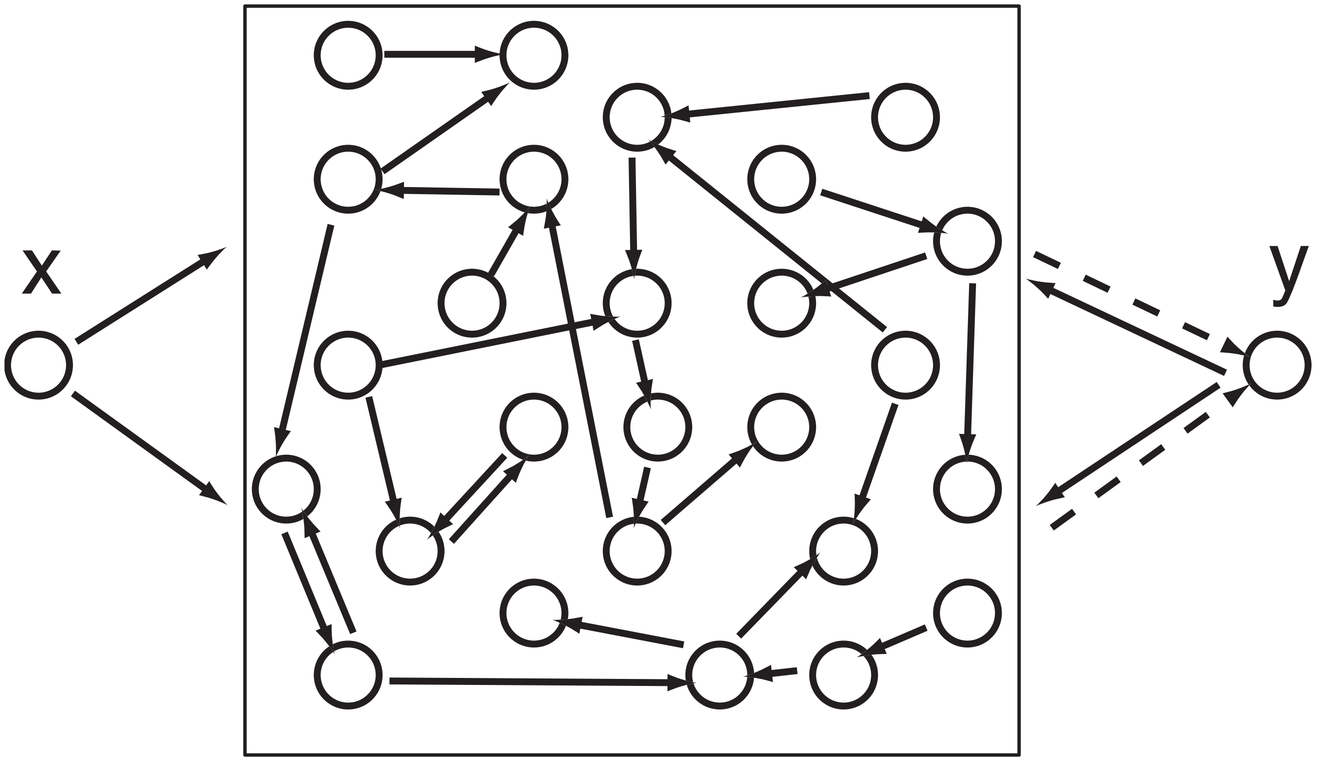
Architectures



a) feed-forward
b) feedforward multilayer
c, d) recurrent
e) fully connected

Reservoir Architectures

Exploit rich dynamics in the reservoir of hundreds of randomly interconnected neurons with low connectivity (0.01, e.g)



Liquid State Machines (Maas et al, 2002)
Echo State Networks (Jaeger et Haas, 2004)

Local vs Distributed Input Encoding

1	●	○	○	○	○
2	○	●	○	○	○
3	○	○	●	○	○
4	○	○	○	●	○
5	○	○	○	○	●

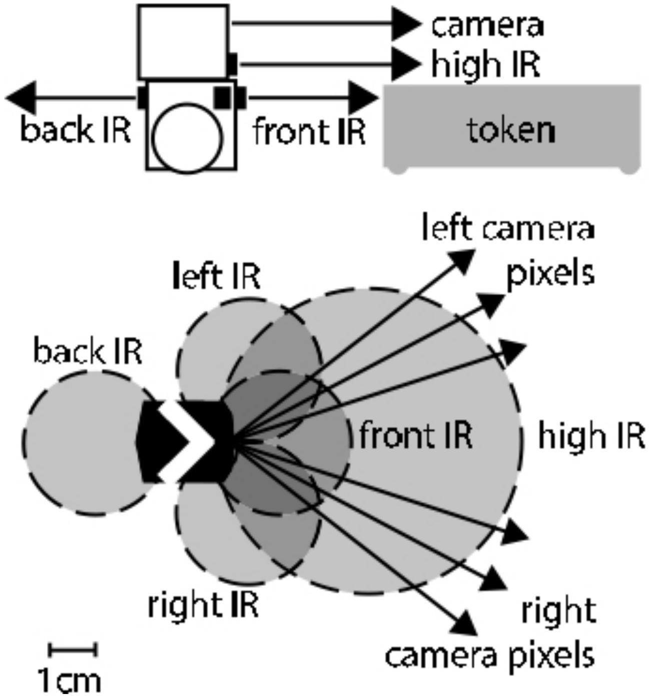
LOCAL
One neuron stands for one item
Extremely sparse coding
Grandmother cells
Scalability problem
Robustness problem

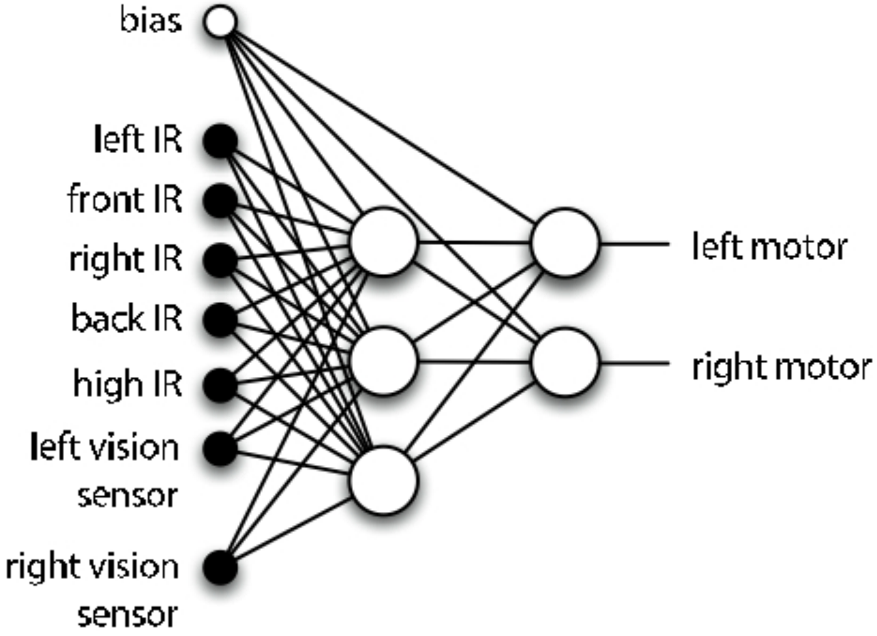
1	●	○	○	○	●
2	●	●	○	●	○
3	○	○	●	○	●
4	●	●	●	○	●
5	○	○	○	○	○

DISTRIBUTED
Neurons encode features
One neuron may stand for >1 item
One item may activate >1 neuron
Robust to damage
Sparsity can vary

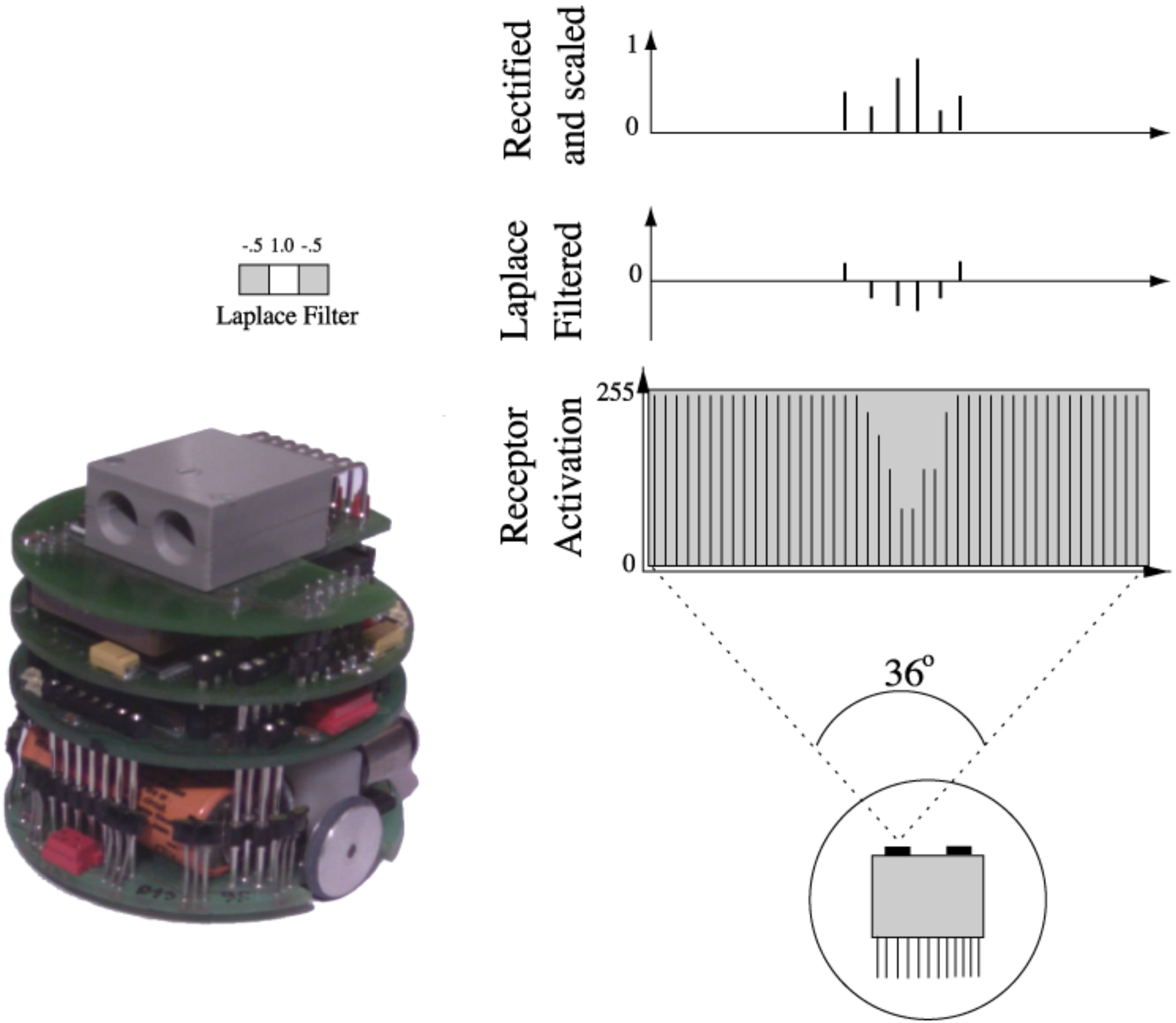
Input Normalisation

$$x_i' = \frac{x_i}{\sqrt{\sum_{j=1}^{N} x_j^2}}$$





Contrast Detection

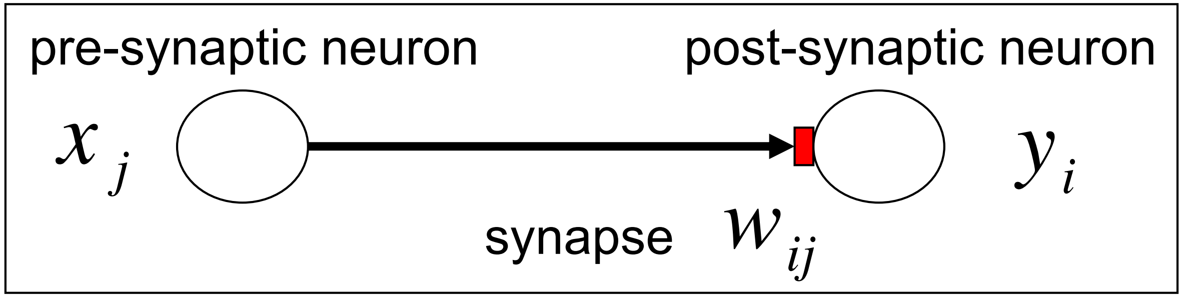


sparse coding

dense coding

Learning

Learning is experience-dependent modification of connection weights



$$\Delta w_{ij} = x_j y_i$$

Hebb’s rule (1949)

Learning is a gradual process and requires many input-output comparisons

Learning cycle

1. Initialize weights (e.g., random values from normal distribution)
2. Present randomly selected input pattern to network
3. Compute values of output units
4. Compute weight modifications
5. Update weights

Standard weight update

$$w_{ij}^{t} = w_{ij}^{t-1} + \eta \Delta w_{ij}$$

learning rate [0,1]

6. Repeat from 2. until weights do not change anymore

Learning modalities

Unsupervised learning

Supervised learning

Reinforcement learning

Evolutionary (learning)

Unsupervised learning

The weight change depends only on the activity of the pre-synaptic and of the postsynaptic neurons

$$\Delta w_{ij} = x_j y_i$$

Unsupervised learning is used for

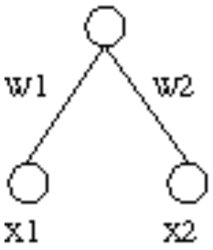
- Detecting statistical features of the input distribution
- Information compression and reconstruction
- Discovery of topological relationships in the input data
- Memorization

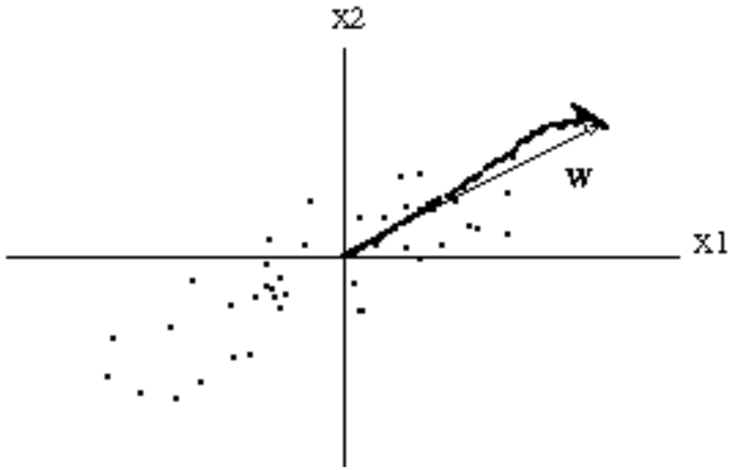
Oja's learning rule

Hebb's rule suffers from **self-amplification** (unbounded growth of weights)**.** Biological synapses cannot grow indefinitely

Oja (1982) introduced self-limiting growth factor in Hebb rule

$$\Delta w_j = \eta y\left(x_j - w_j y\right)$$





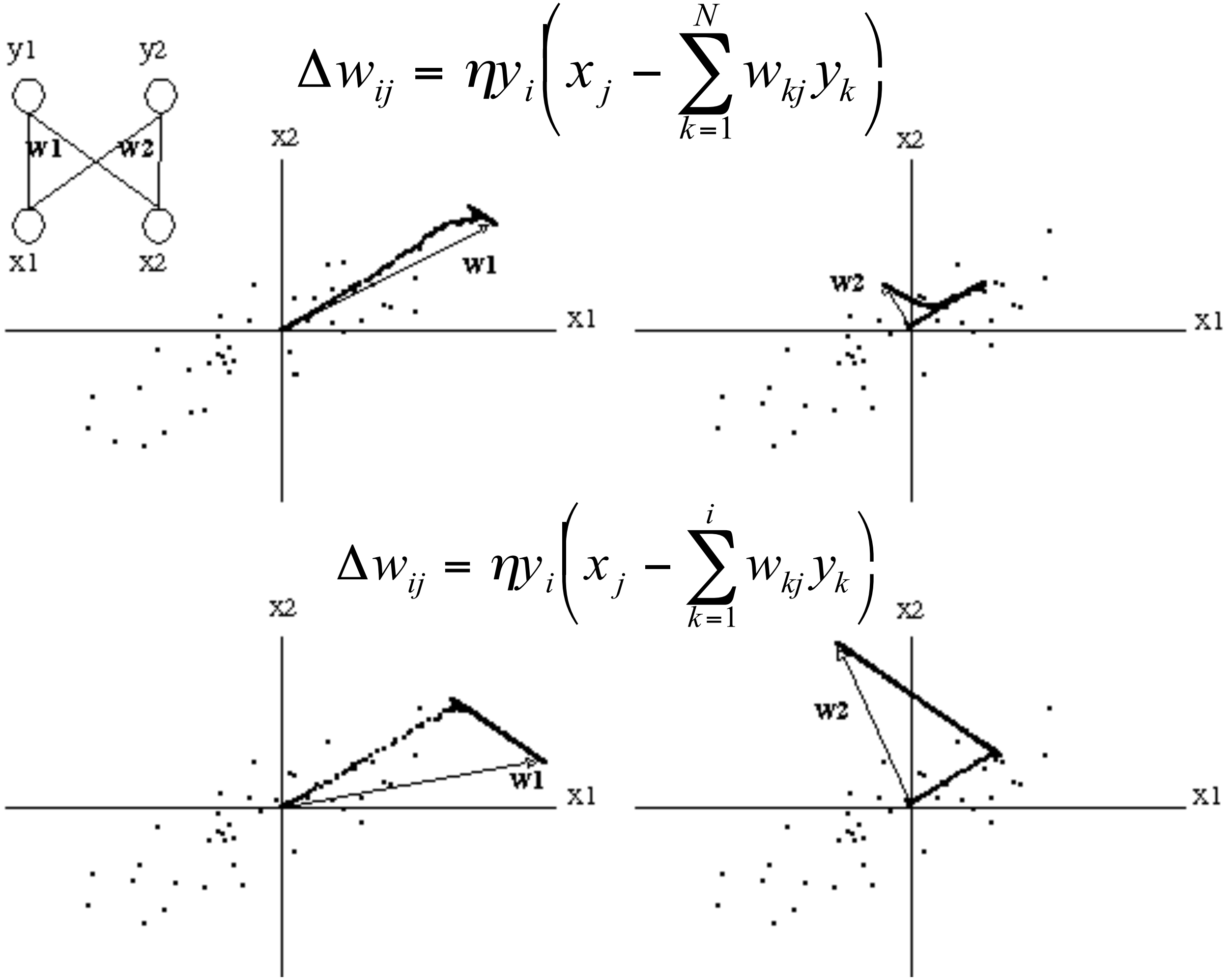
As a result, the weight vector develops along the direction of maximal variance of the input distribution.

Neuron learns **how familar** a new pattern is: input patterns that are closer to this vector elict stronger response than patterns that are far away.

Principal Component Analysis

Oja rule for N output units develops weights that span the sub-space of the N principal components of the input distribution.

$$\Delta w_{ij} = \eta y_i \left(x_j - \sum_{k=1}^{N} w_{kj} y_k \right)$$

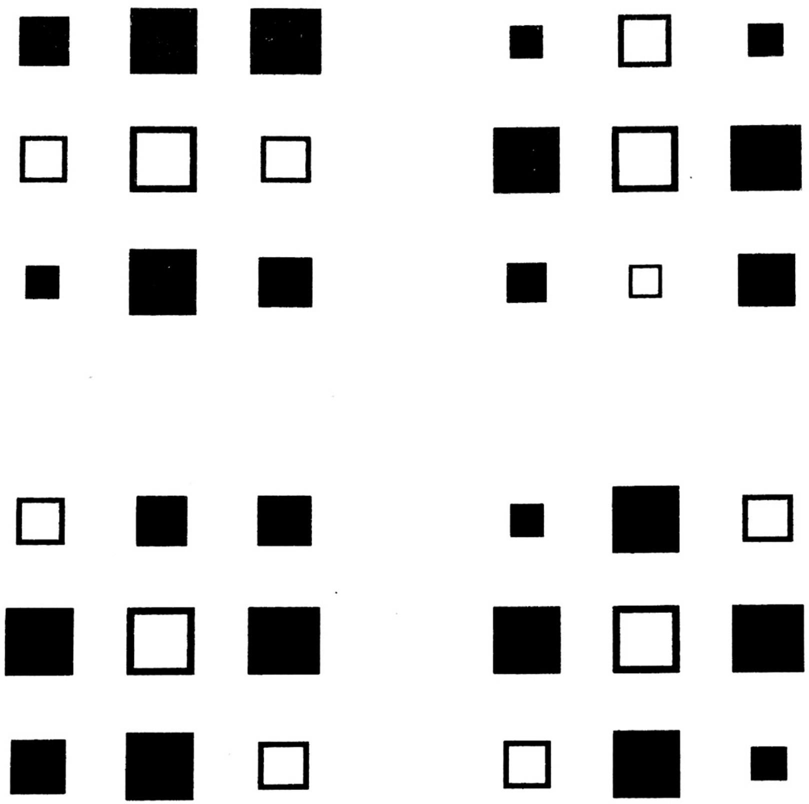


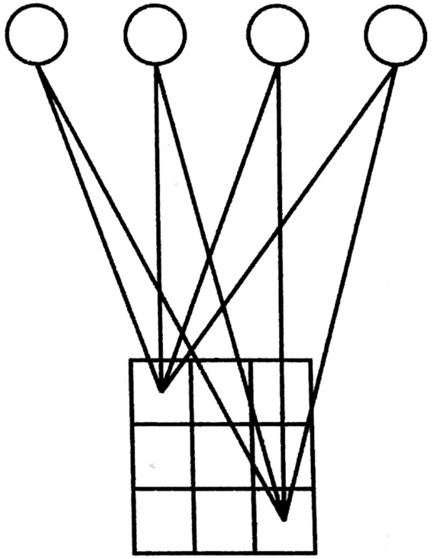
Sanger rule for N output units develops weights that correspond to the N principal components of the input distribution.

$$\Delta w_{ij} = \eta y_i \left(x_j - \sum_{k=1}^{i} w_{kj} y_k \right)$$

Receptive Fields

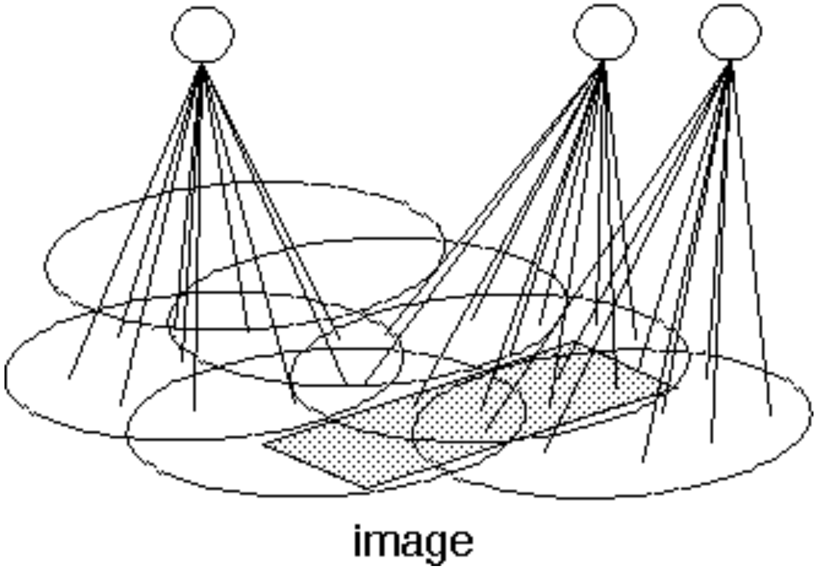
The **Receptive Field** indicates the input area subtended by a neuron *and* the input pattern that generates the strongest activation.
RF can be visualized by plotting the weight pattern in the input space.

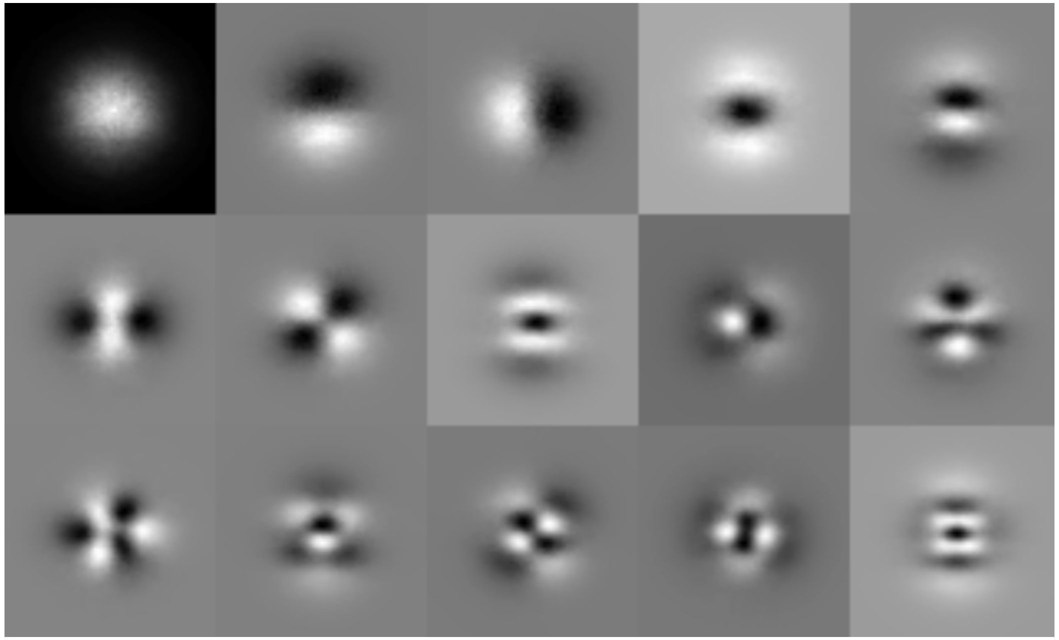




Do brains compute PCA?

An Oja network with multiple output units exposed to a large set of natural images develops receptive fields similar to those found in the visual cortex of all mammals [Hancock et al., 1992]



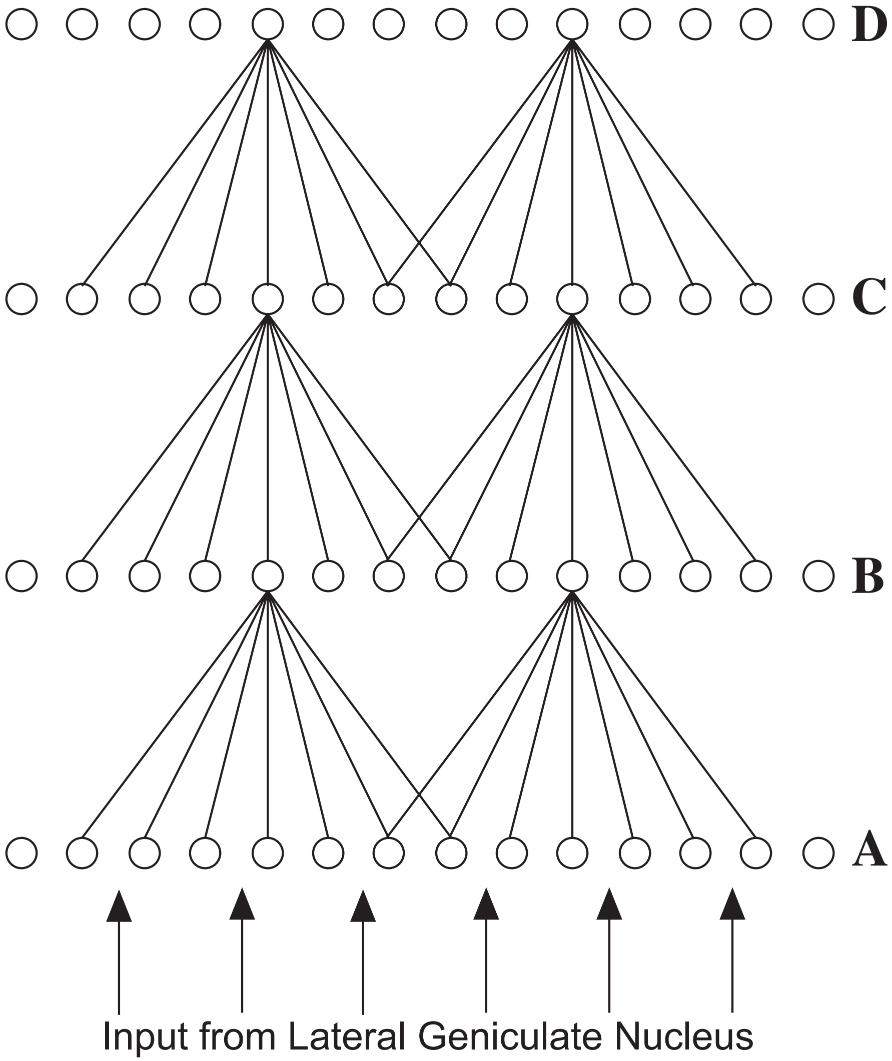


Mammals are born with pre-formed hierarchically-organized feature detectors. But they never saw anything in the womb: how can it be?

Image: Kelly Frankenburg

Multilayer Feature Detection

Linsker (1986)



Topologically restricted connectivity

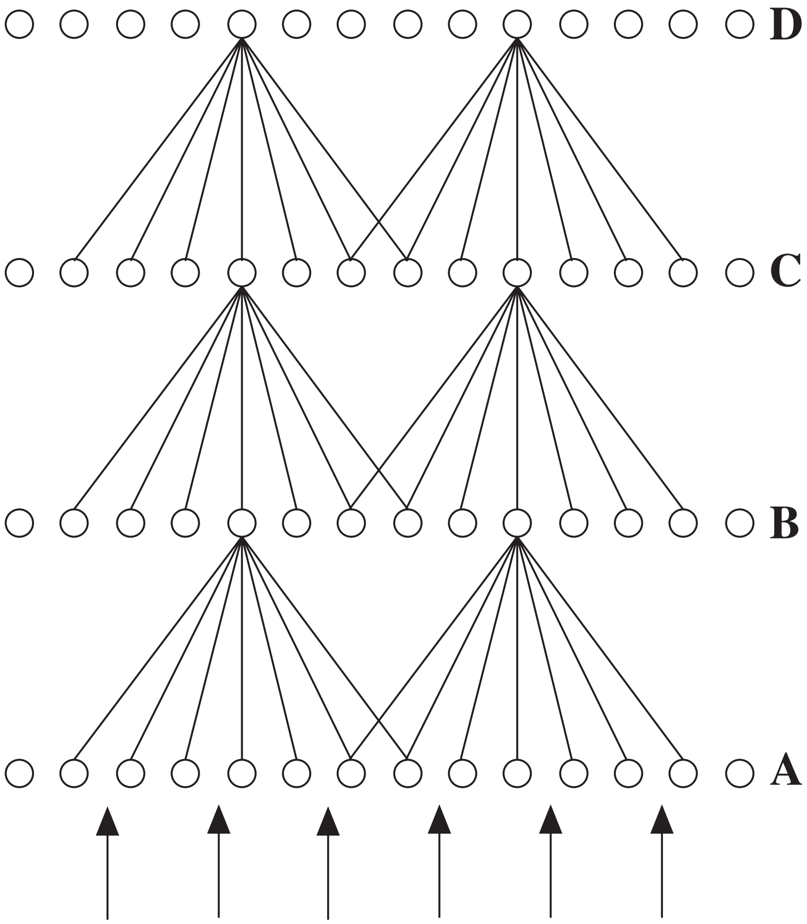
Linear activation function

Plain Hebbian learning with weight clipping at *w*+ and *w*-

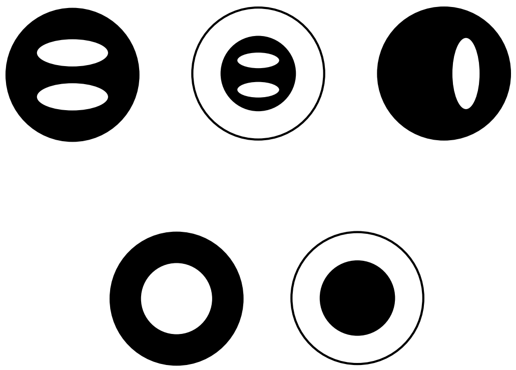
Learn one layer at a time, starting from lower layer

Emerging Receptive Fields

Linsker (1986)

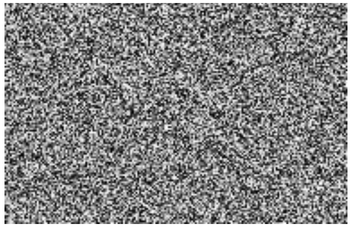


Input from Lateral Geniculate Nucleus



Average luminosity in RF

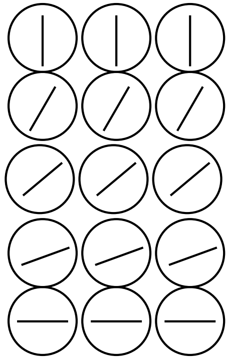
+++++++++



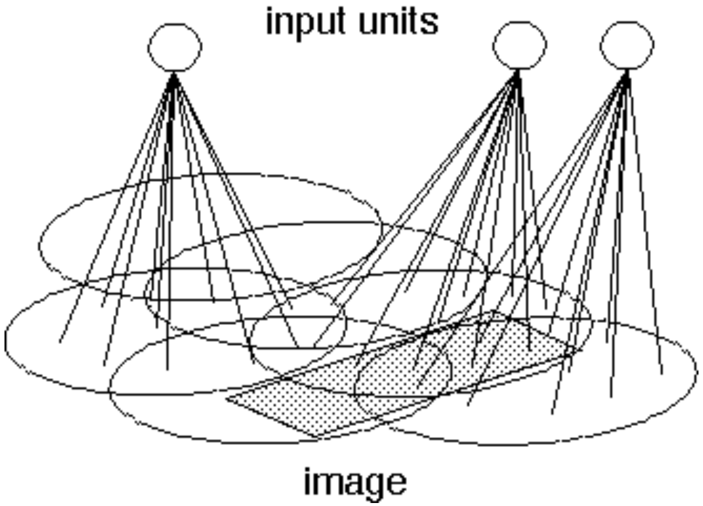
Sensory maps

Neighbouring neurons respond to similar patterns with gradual transitions

The visual cortex is organized in specialized modules. Each module is composed by a series of columns of neurons. Neurons respond to bars of different orientation

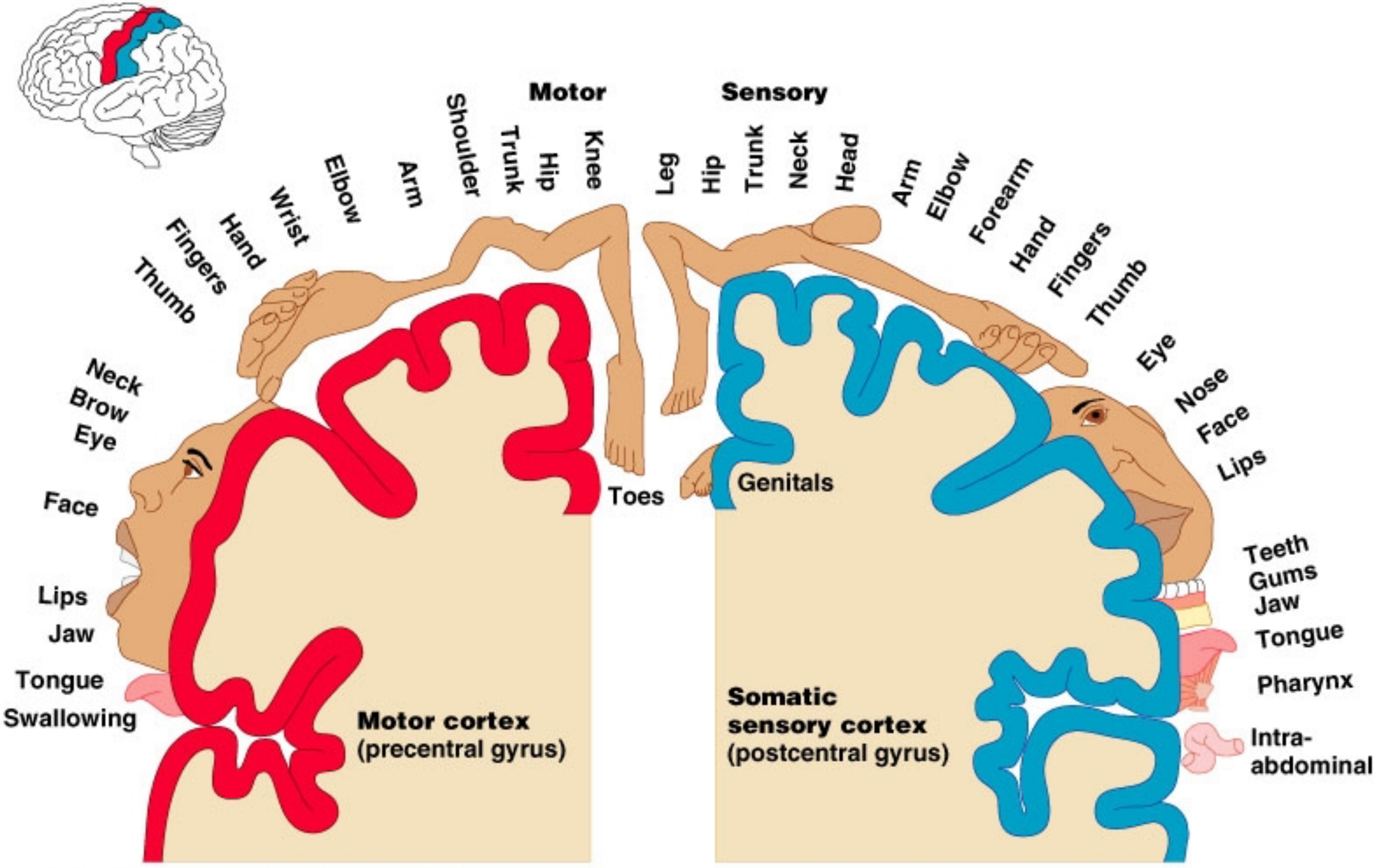


1. The bar orientation gradually varies along the column.
2. Neighbouring columns correspond to neighbouring areas of the retina (**retinotopic maps**).



A similar structure exists in the auditory cortex (**tonotopic maps**).

Sensory-Motor Body Map

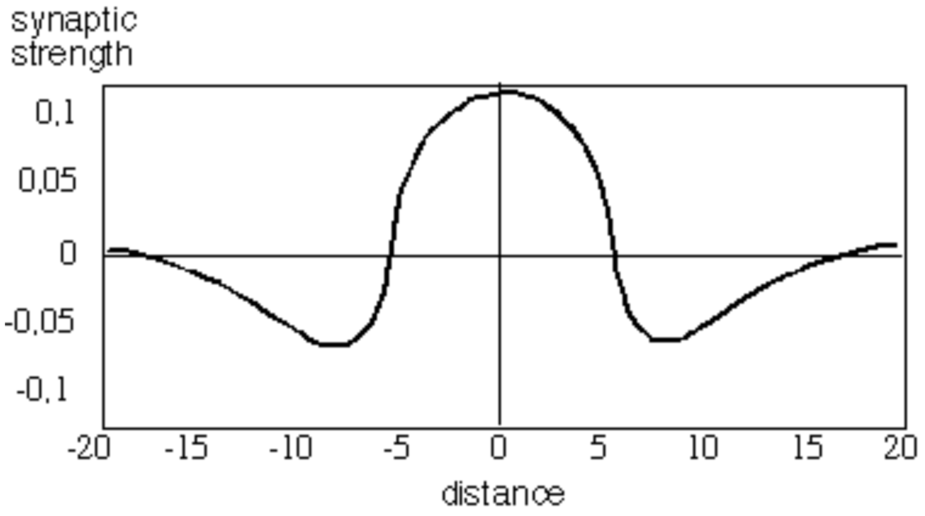


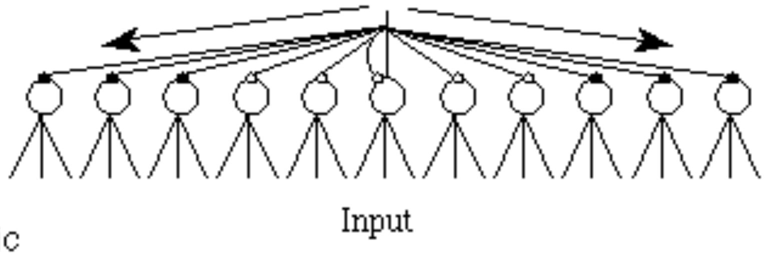
Lateral cortical connections

Cortical neurons display the following pattern of *projective* connectivity:

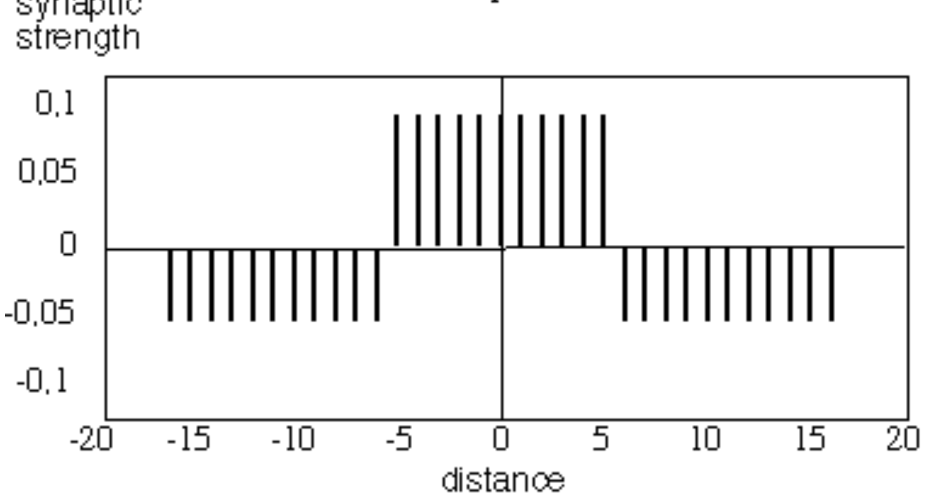
- up to 50-100 μm radius = excitatory
- up tp 200-500 μm radius = inhibitory
- up to few cm radius = slightly excitatory

It resembles to a *Mexican Hat*

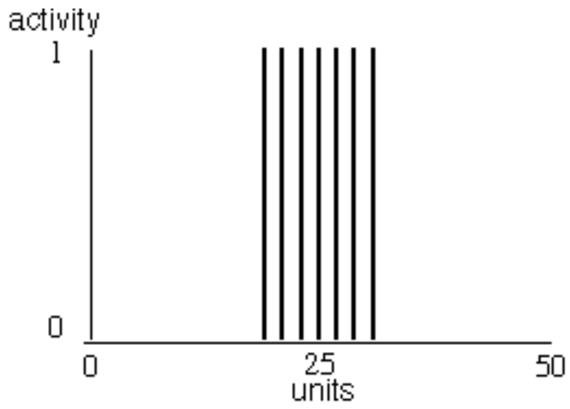




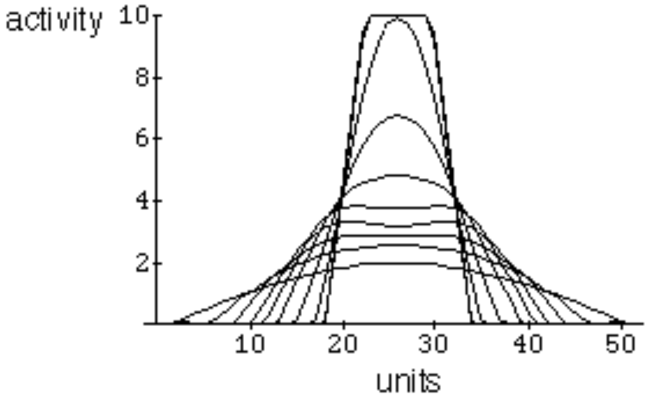
In a neural network, we can approximate the Mexican hat to a bipolar weight distribution.



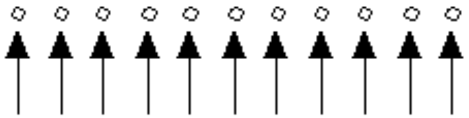
Neural bubble formation



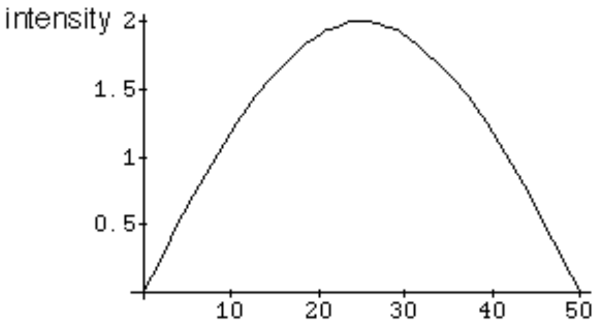
Simplification: directly set output of unit with highest activation and its *n* neighbors to 1, and all other units to 0



Gradual emergence of bubble centered around unit with strongest activation



Laterally connected neurons



Input pattern

Self-Organizing Maps

Kohonen (1982)

Let’s apply Hebb rule to layer of such laterally connected neurons

$$y_i = \Phi(A_i) = \begin{cases} 1 & \text{if within neighbourhood } \Psi(y) \\ 0 & \text{otherwise} \end{cases}$$

$$\Delta w_{ij} = \eta y_i (x_j - \Psi(y_i) w_{ij}) \qquad \Psi(y_i) = \begin{cases} \psi & \text{if } y_i = 1 \\ 0 & \text{if } y_i = 0 \end{cases}$$

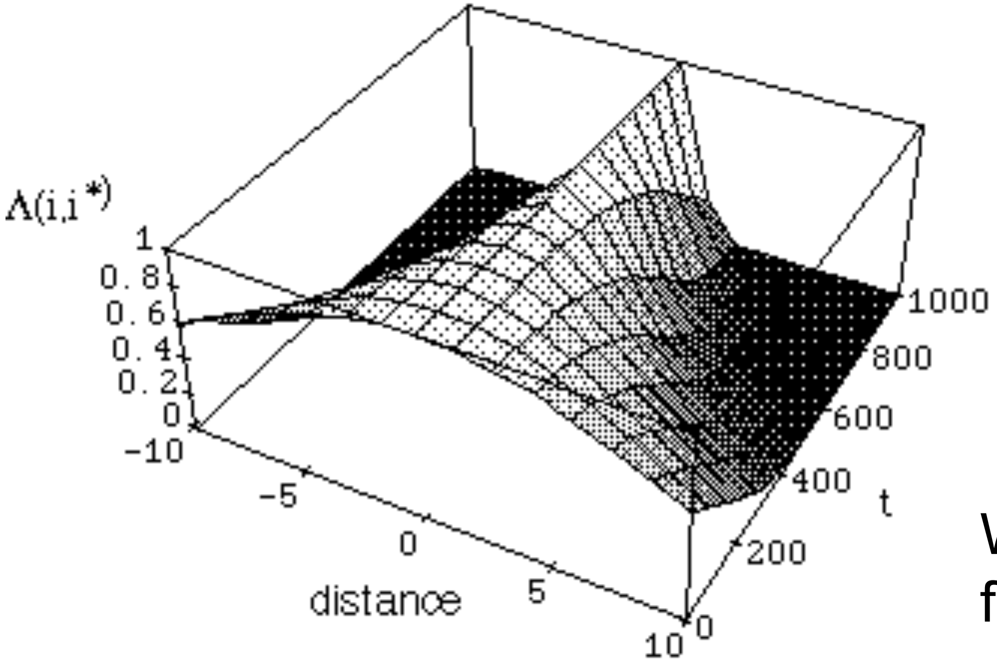
If we set $\Psi(y_i)$ equal to the learning rate η, then the learning rule becomes:

$$\Delta w_{ij} = \begin{cases} \eta(x_j - w_{ij}) & \text{if } y_i = 1 \\ 0 & \text{if } y_i = 0 \end{cases} \quad \text{and} \quad \mathbf{w}_i^{t+1} = \begin{cases} \mathbf{w}_i^t + \eta(\mathbf{x} - \mathbf{w}_i^t) & \text{if } y_i = 1 \\ \mathbf{w}_i^t & \text{if } y_i = 0 \end{cases}$$

1. The weights are changed only for the neurons that are geographically near the neuron with the highest activity,
2. The change moves the weight vector towards the input pattern.

Neighborhood function

The neighbourhood size $\Psi(y)$ is a critical aspect of map self-organization. It should be large at the beginning of training to give a chance to all neurons to change weights and gradually shrink

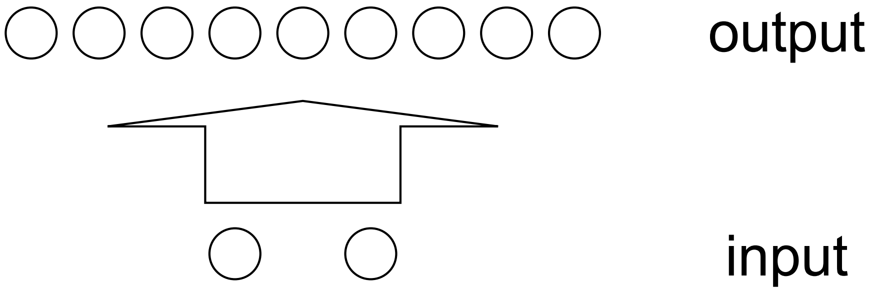


$$\Lambda\left(i,i^*\right)=\begin{cases}1 & \text{if } \left\|\mathbf{c}_i-\mathbf{c}_{i^*}\right\|<=r\\ 0 & \text{otherwise}\end{cases}$$

We can incorporate the neighborhod function in the SOM learning rule

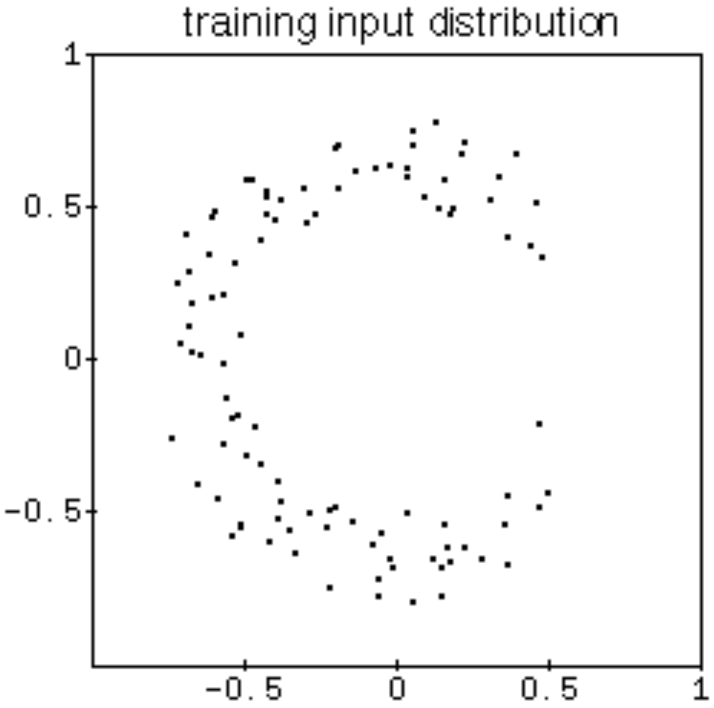
$$\Delta w_{ij}=\eta\Lambda(i,i^*)\left(x_j-w_{ij}\right)$$

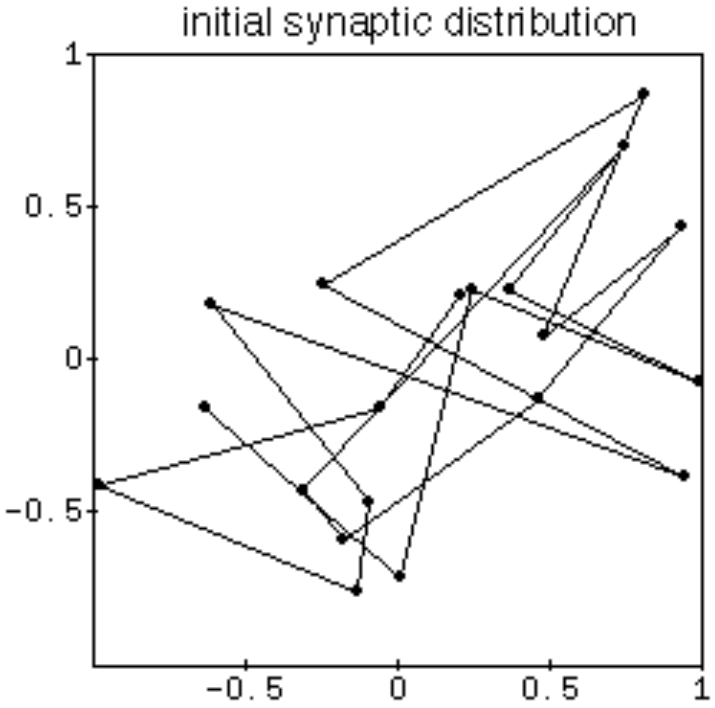
Example of self-organizing map



output

input

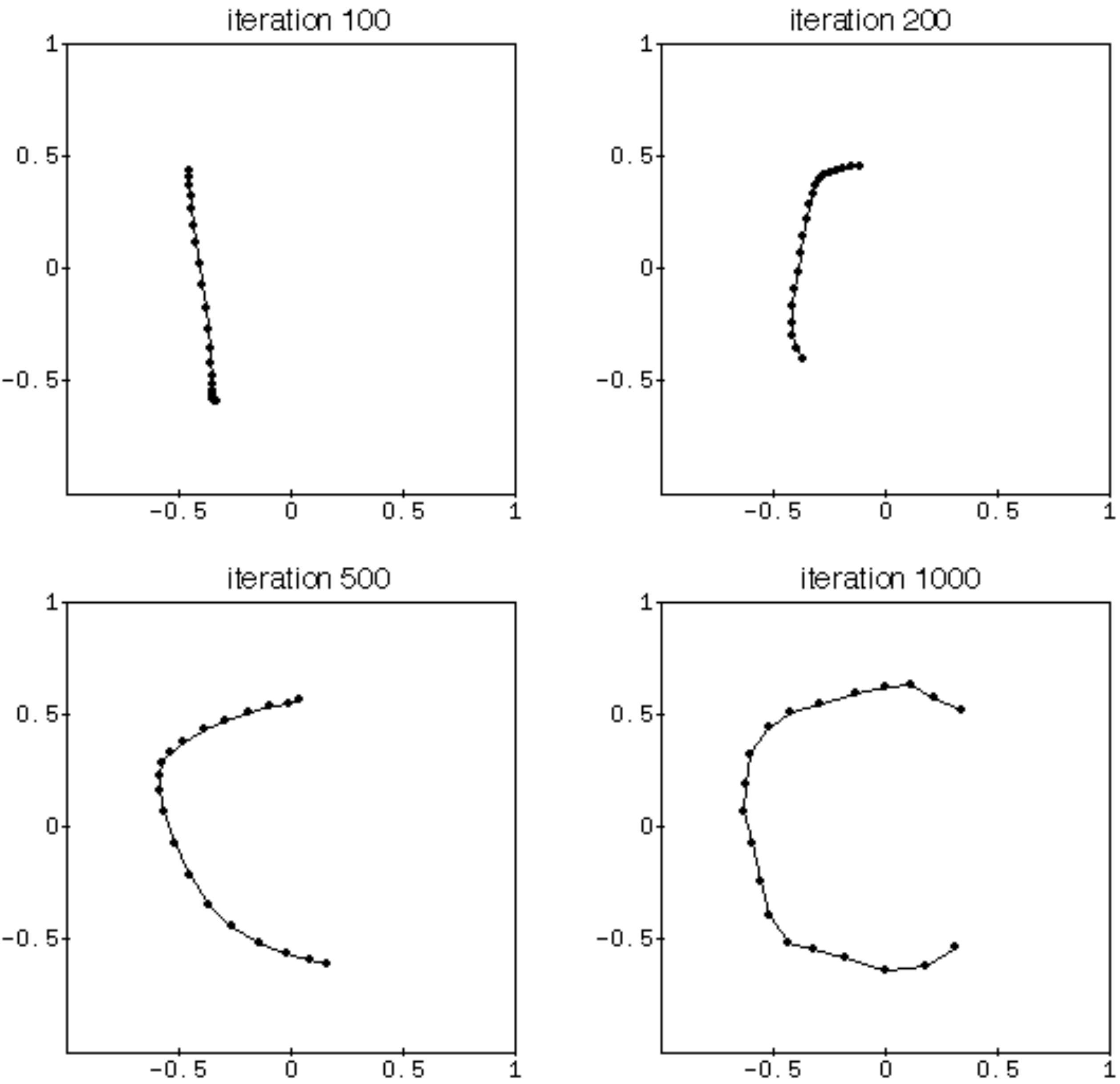

training input distribution


initial synaptic distribution

Self-organization phases

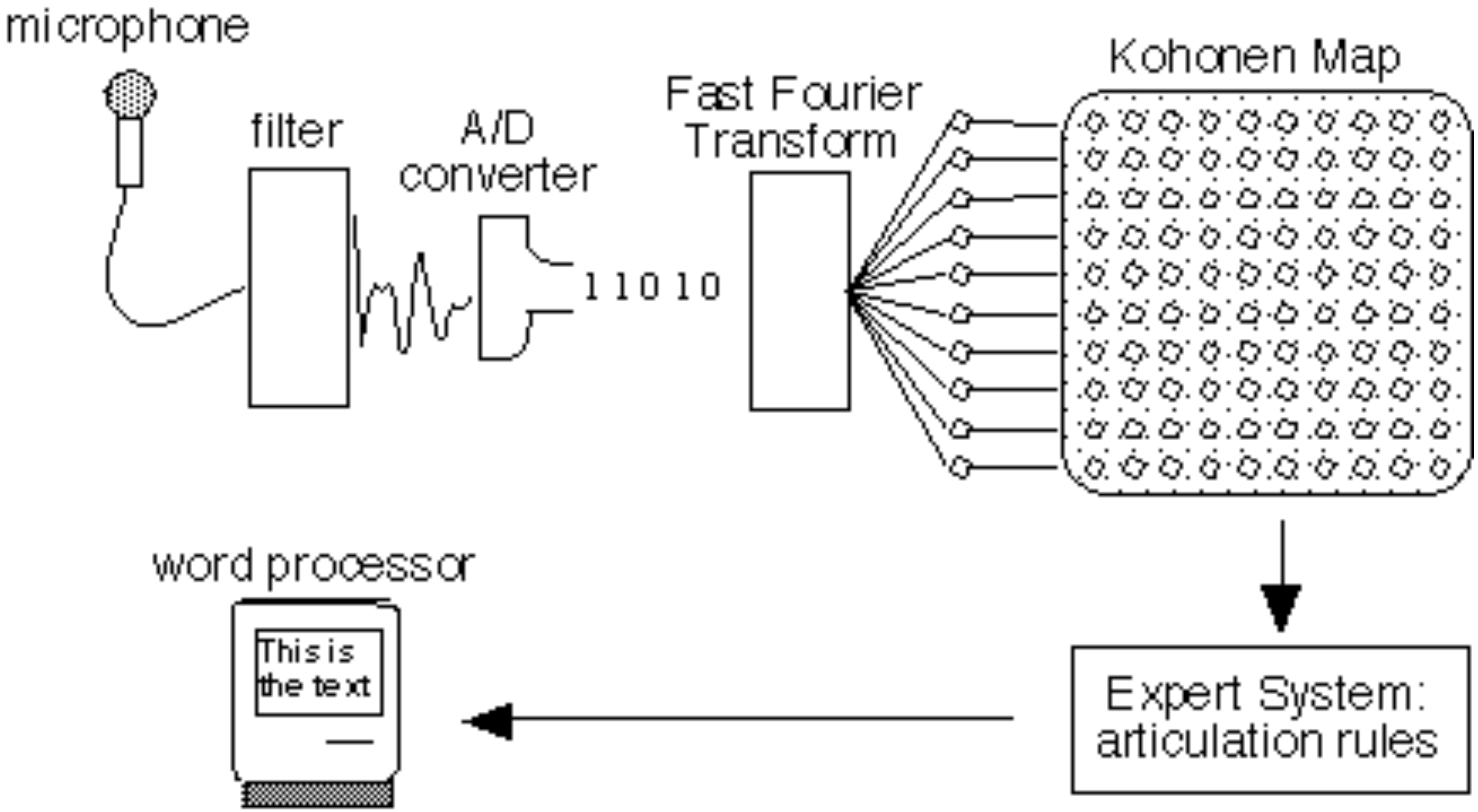
Ordering phase:
Fast
Neighborhood change

Convergence phase:
Slow
No neighborhood change



The phonetic typewriter

The phonetic typewriter transform a spoken text into written text.
Output layer provides a compact representation of speech.
Topological representations helps discrimination of similar phonemes.
It requires manual labeling and articulation rules


microphone
filter
A/D converter
1 1 0 1 0
Fast Fourier Transform
Kohonen Map
word processor
This is the text
Expert System: articulation rules