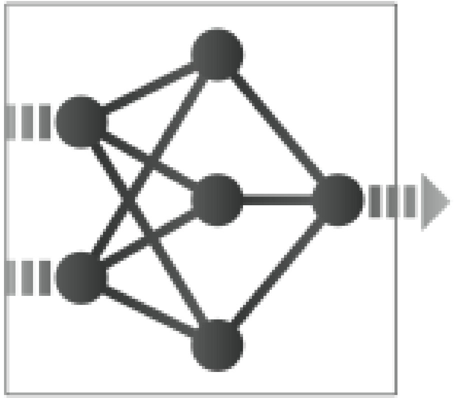
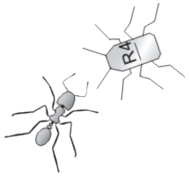
Neural Systems (1)





Why Nervous Systems?

Not all animals have nervous systems; some use only chemical reactions
Paramecium and sponge move, eat, escape, display habituation

QuickTime™ and a
TIFF (LZW) decompressor
are needed to see this picture.

Nervous systems give **advantages**:

1) Selective transmission of signals across distant areas (=more complex bodies)
2) Complex adaptation (=survival in changing environments)

Biological Neurons

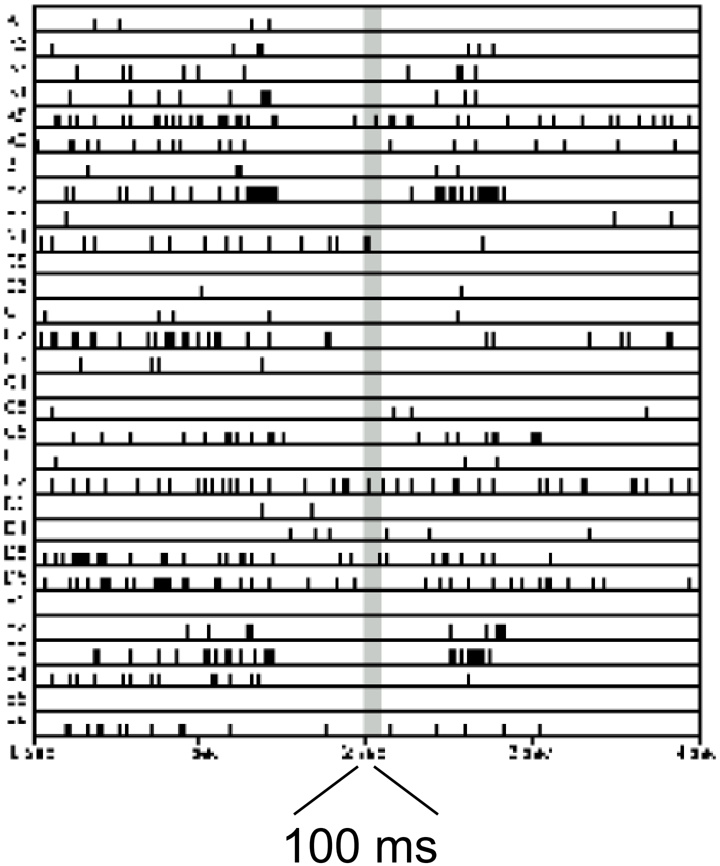
QuickTime™ and a
TIFF (LZW) decompressor
are needed to see this picture.

Type of Neurons

QuickTime™ and a
TIFF (LZW) decompressor
are needed to see this picture.

Interneurons can be
1- Excitatory
2- Inhibitory

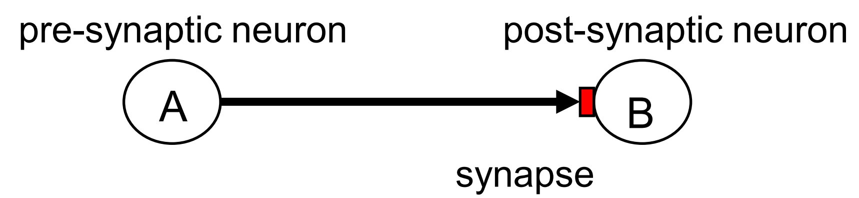
How Do Neurons Communicate?



Firing rate → McCulloch-Pitts → Connectionism

Firing time → Spiking neurons → Computational Biology

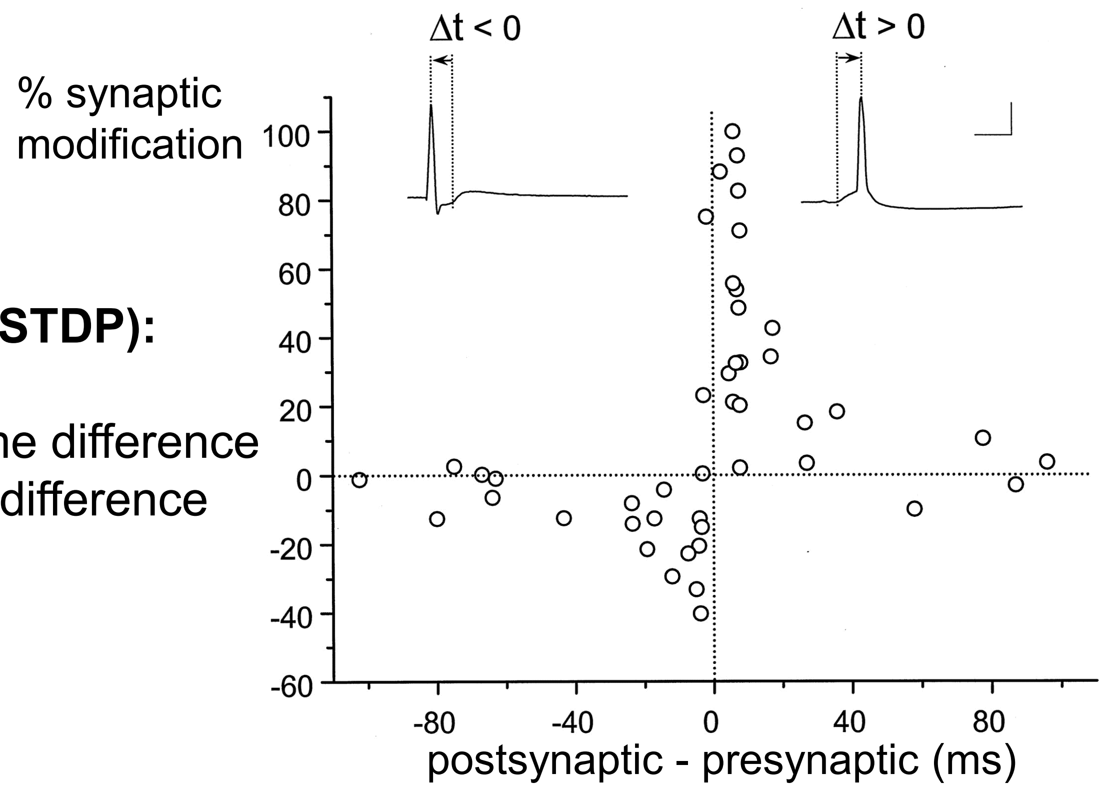
Synaptic Plasticity



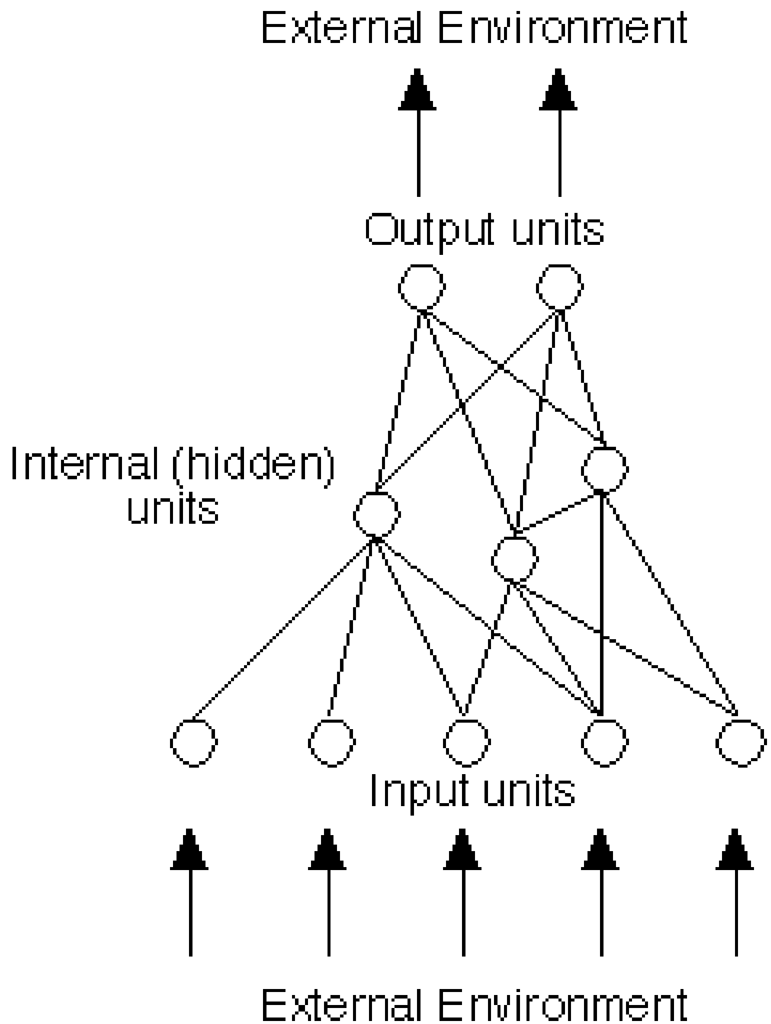
Hebb rule (1949):
Synaptic strength is increased if cell A consistently contributes to firing of cell B

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



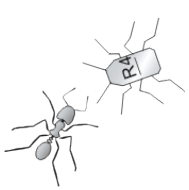
An Artificial Neural Network



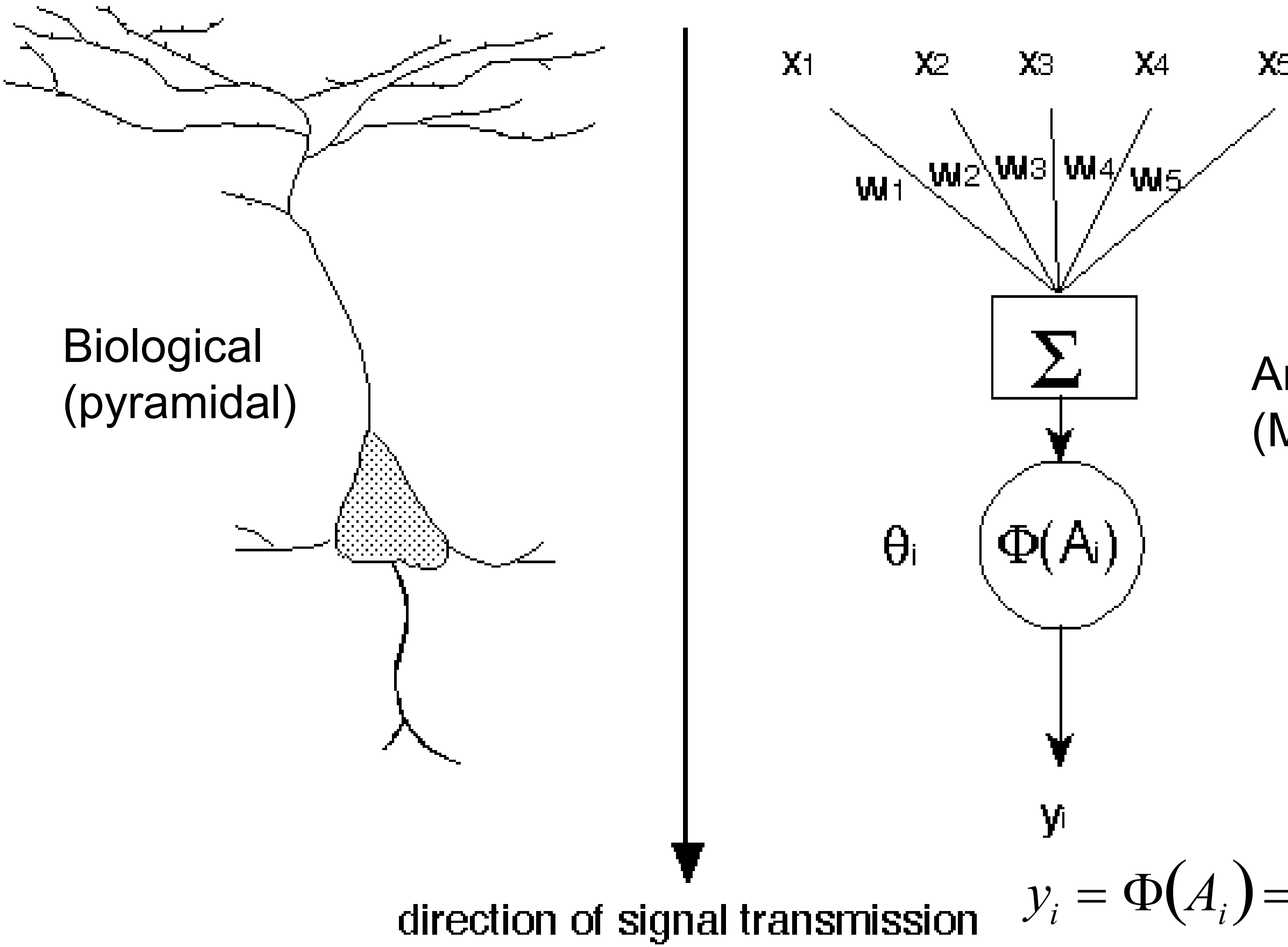
A neural network communicates with the environments through input units and output units. All other elements are called internal or hidden units.

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

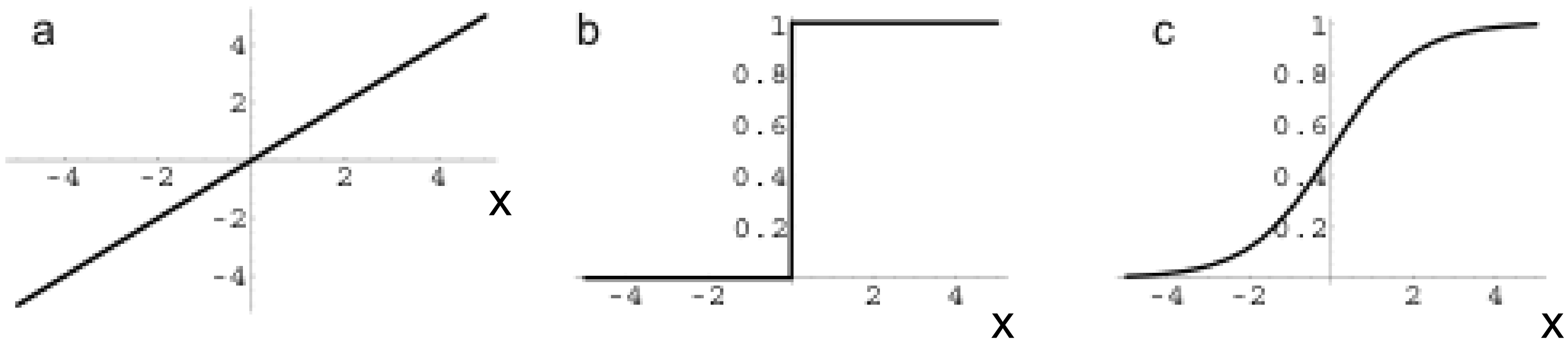


Artificial
(McCulloch-Pitts)

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

Output functions

Identity $\Phi(x)$ | Step $\Phi(x)$ | Sigmoid $\Phi(x)$



Sigmoid function:
- continuous
- non-linear
- monotonic
- bounded
- asymptotic

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

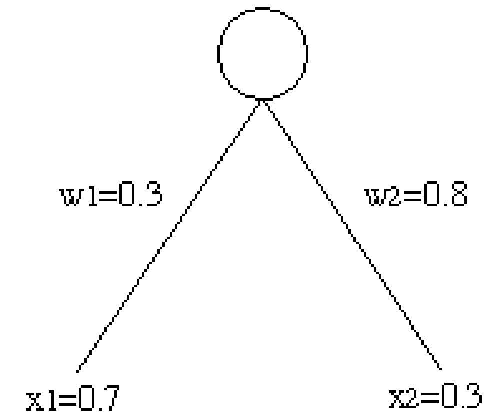
$$\Phi(x) = \tanh(kx)$$

Signalling Input Familiarity

The output of a neuron is a measure of how similar is its current input pattern to 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:

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

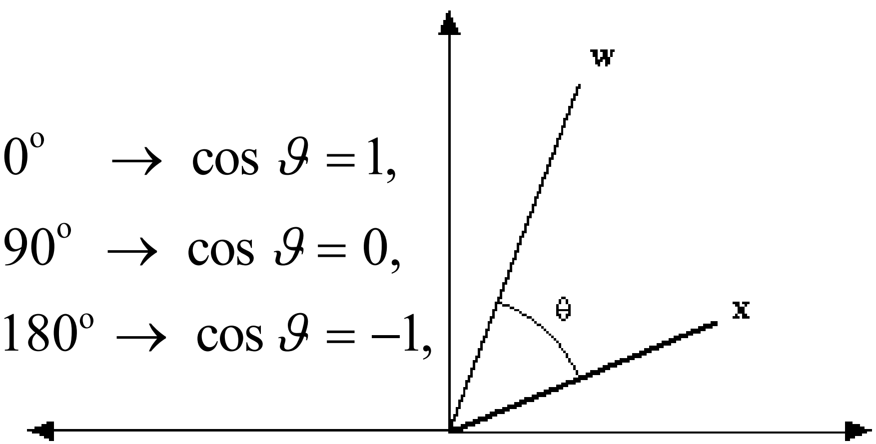
where the vector length is:

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

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

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

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



3. Output signals input familiarity

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

Separating Input Patterns

A neuron divides the input space in two regions, one where A>=0 and one where A<0.
The separation line is defined by the synaptic weights:

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

QuickTime™ and a
TIFF (LZW) decompressor
are needed to see this picture.

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

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.

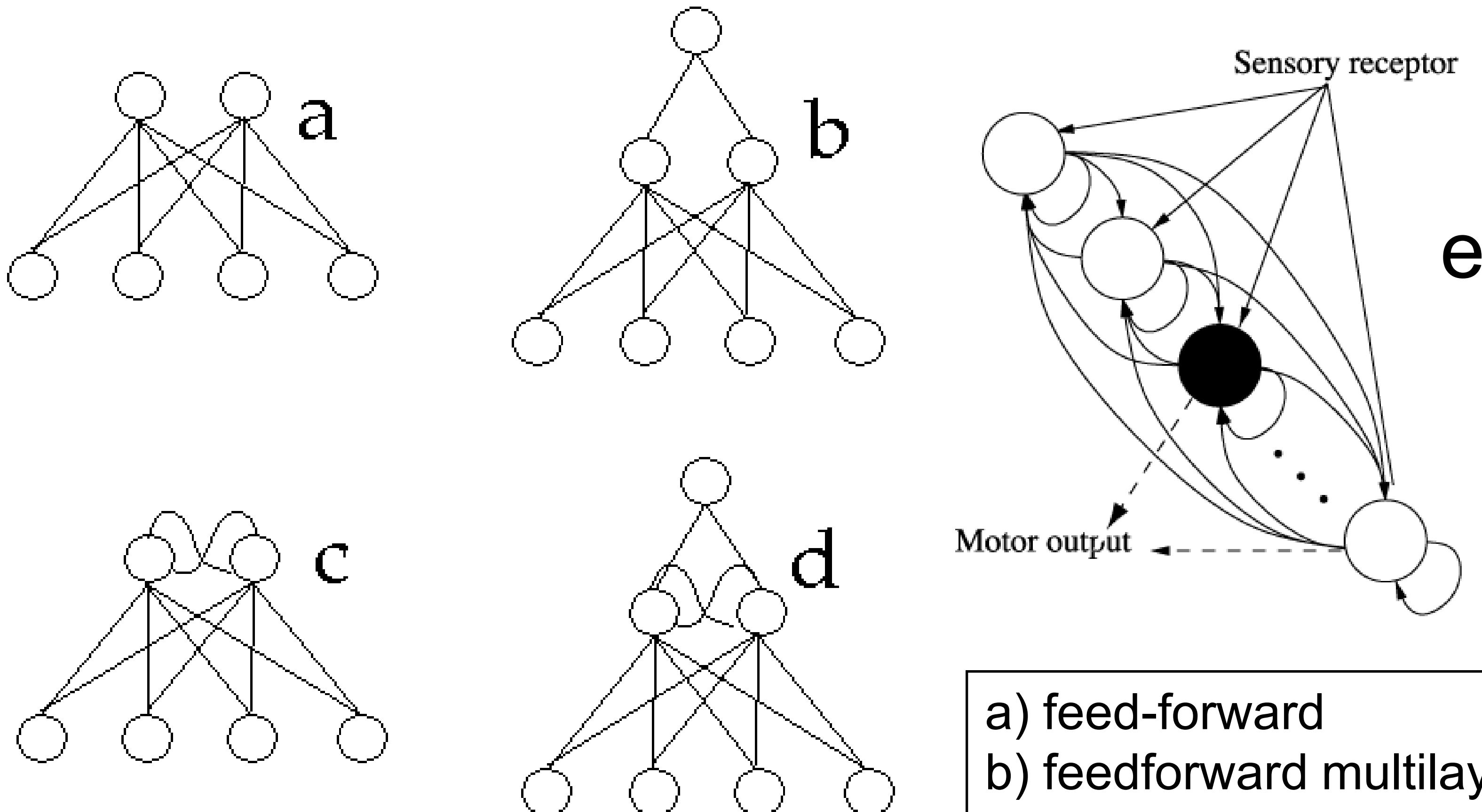
$$y_i = \Phi(A_i) = \Phi\left(\sum_{j=1}^{N} w_{ij} x_j - \vartheta_i\right)$$

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

QuickTime™ and a
TIFF (LZW) decompressor
are needed to see this picture.

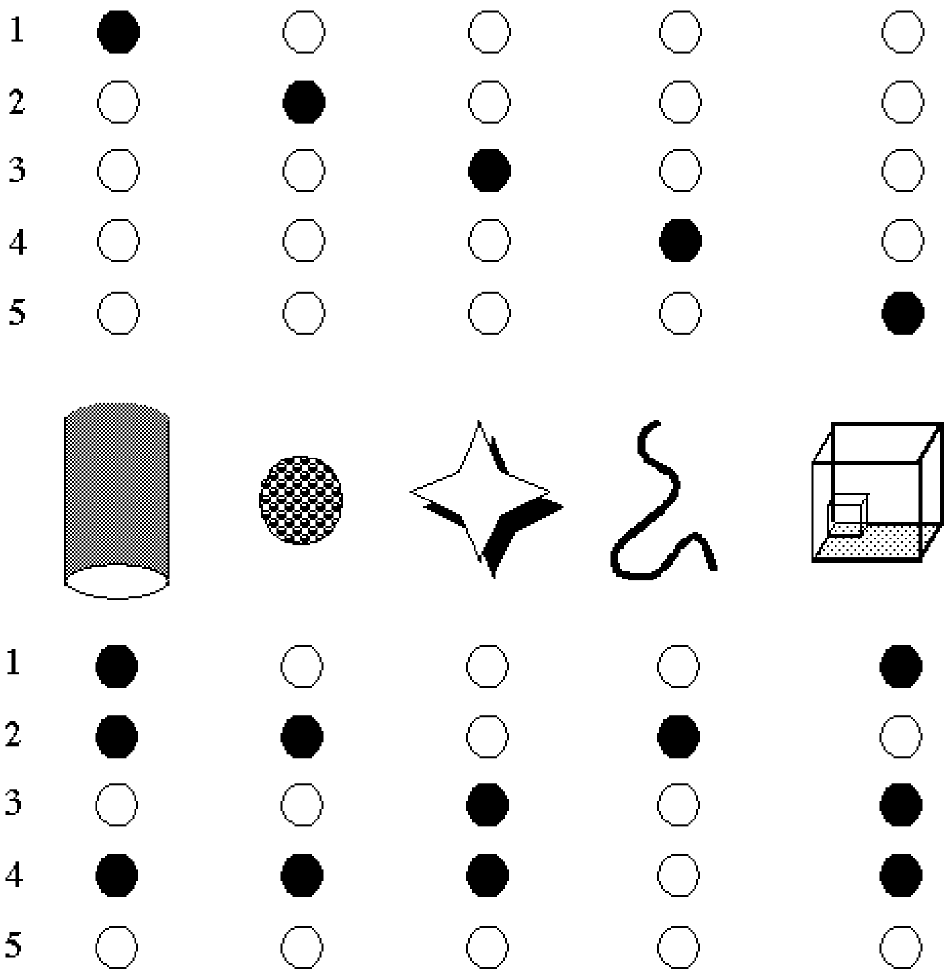
- Easier to express/program
- Threshold is adaptable like other weights

Architectures



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

Input Encoding



LOCAL

One neuron stands for one item

Grandmother cells

Scalability problem

Robustness problem

DISTRIBUTED

Neurons encode features

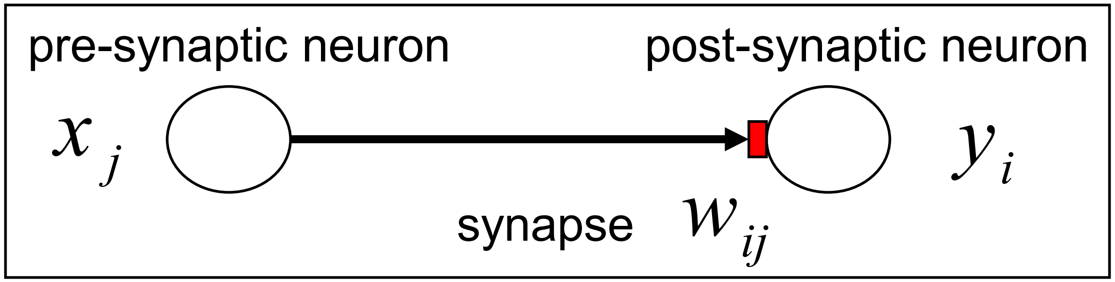
One neuron may stand for >1 item

One item may activate >1 neuron

Robust to damage

Learning

Learning is experience-dependent modification of connection weights



Hebb's rule (1949) $$\Delta w_{ij} = x_j y_i$$

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

learning rate
(in the range [0,1])

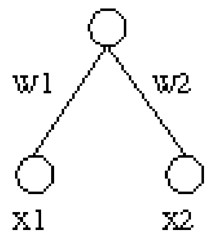
Hebb's rule suffers from **self-amplification** (unbounded growth of weights)

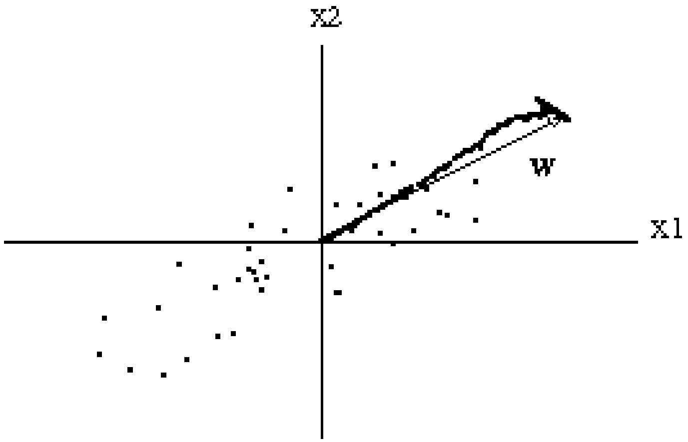
Unsupervised Learning

Biological synapses cannot grow indefinitely

Oja (1982) proposed to limit weight growth by introducing a self-limiting factor

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



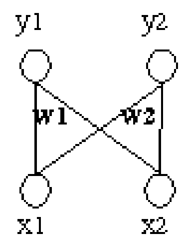


As a result, the weight vector develops along the direction of maximal variance of the input distribution.

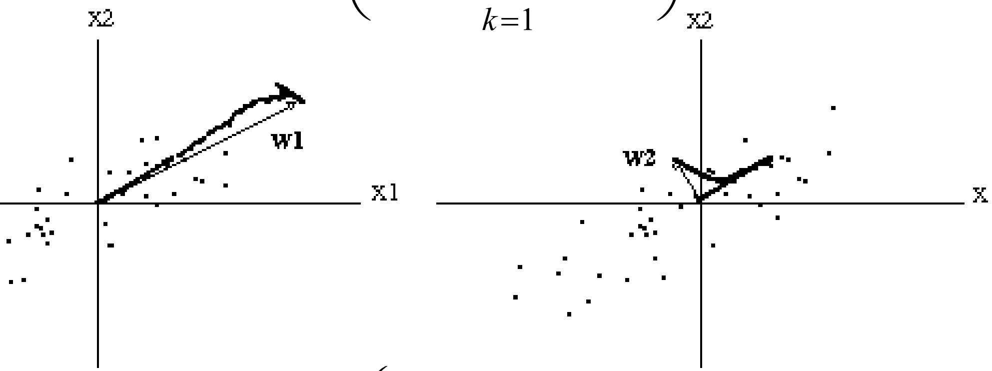
Neuron learns **how familar** a new pattern is: input patterns that are closer to this vector elict stronger response than patterns that are far away.

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.

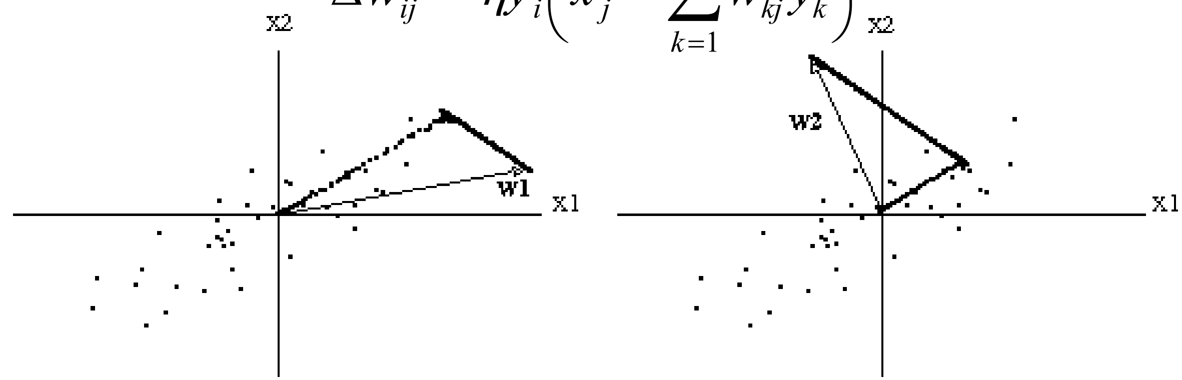


$$\Delta w_{ij} = \eta y_i \left(x_j - \sum_{k=1}^{N} w_{kj} y_k \right)$$



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

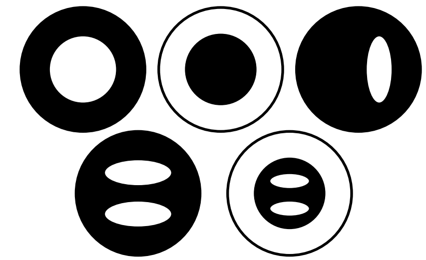
$$\Delta w_{ij} = \eta y_i \left(x_j - \sum_{k=1}^{i} w_{kj} y_k \right)$$



Useful for reduction of dimensionality and feature extraction

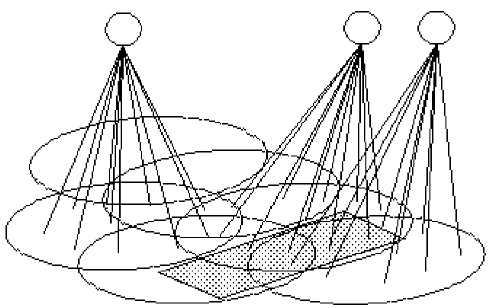
Do brains compute PCA?

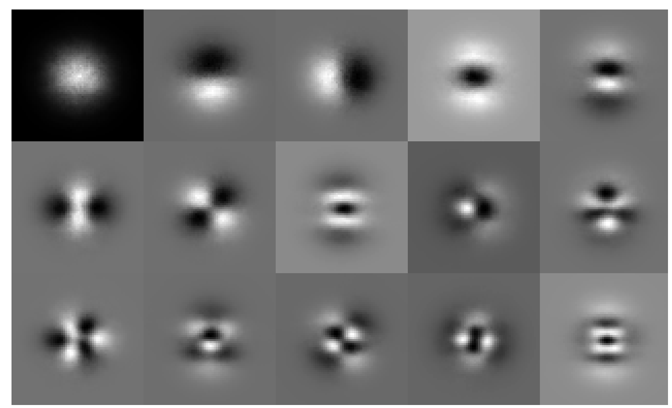
Receptive field is the pattern of stimulation that activates a neuron.
Equivalent to pattern of synaptic weights



Example of visual RF

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]



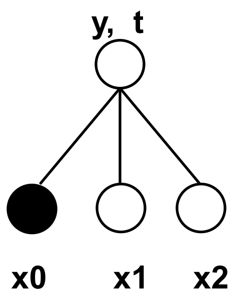


However:
a) PCA cannot detect spatial frequencies, brains do
b) Cannot separate signal sources generated by independent signals

Supervised Learning

- **Teacher** provides desired responses for a set of training patterns
- Synaptic weights are modified in order to reduce the **error** between the output *y* and its desired output *t* (a.k.a. teaching input)

Widrow-Hoff defined the error with the symbol delta: $\delta_i = t_i - y_i$
which is why this learning rule is also known as **delta rule**.



linear units

repeat for every input/output pair until error is 0

$$w_{ij} = rnd(\pm 0.1)$$

initialize weights to random values

$$y_i = \sum_{j=0} w_{ij} x_j$$

present input pattern and compute neuron output

$$\Delta w_{ij} = \eta (t_i - y_i) x_j$$

compute weight change using difference between desired output and neuron output

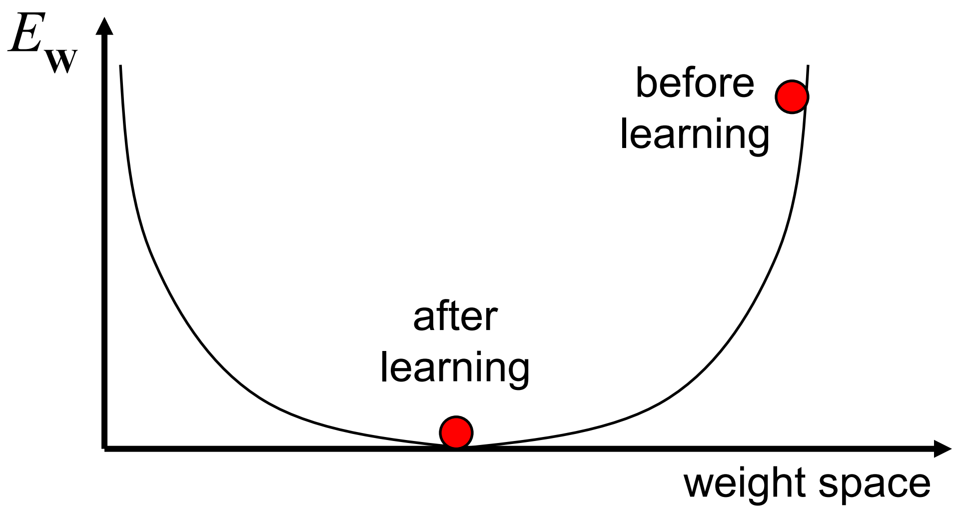
$$w_{ij} = w_{ij}^{t-1} + \Delta w_{ij}$$

get new weights by adding computed change to previous weight values

Error function

The delta rule modifies the weights to descend the gradient of the error function

$$E_{\mathbf{w}} = \frac{1}{2}\sum_{\mu}\sum_{i}\left(t_i^{\mu} - \sum_{j=0} w_{ij}x_j\right)^2$$

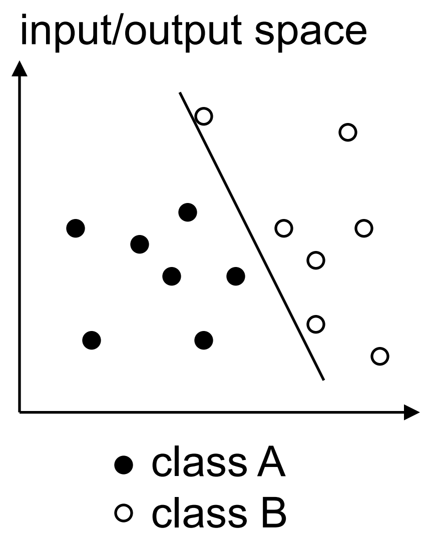


Error function for a network with a single layer of synaptic weights
Network with single layer of weights is also known as **perceptron** (Rosenblatt, 1962)

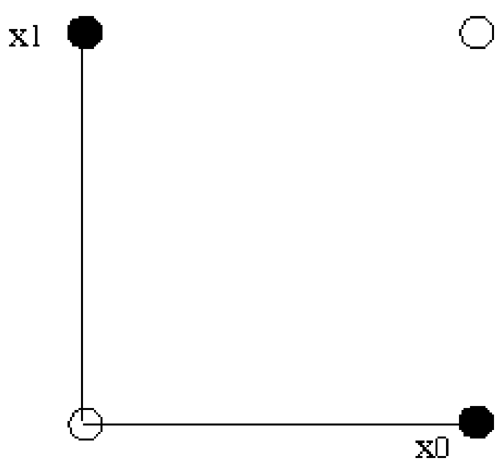
Linear Separability

Perceptrons can solve only problems whose input/output space is **linearly separable**.

Several real world problems are not linearly separable.



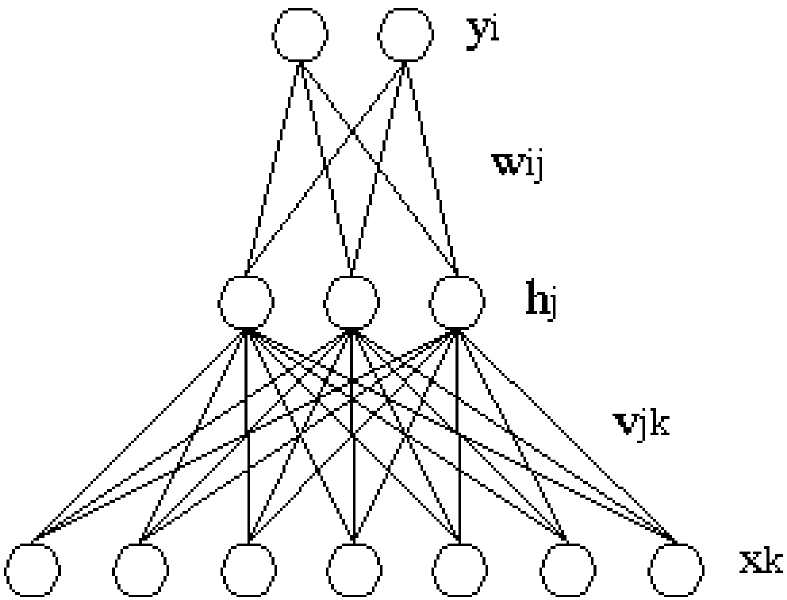
Example of XOR problem



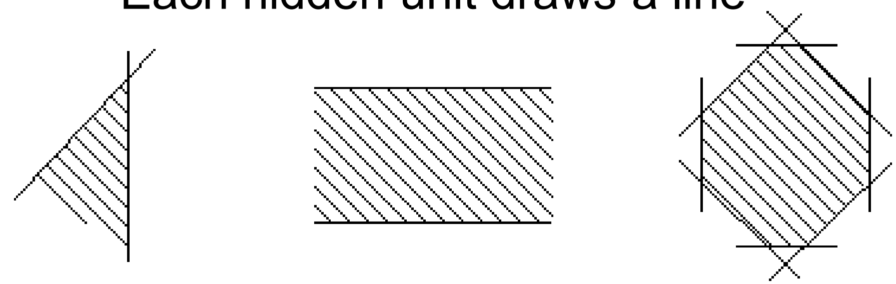
x0	x1	t	
0	0	0	○
1	1	0	○
1	0	1	●
0	1	1	●

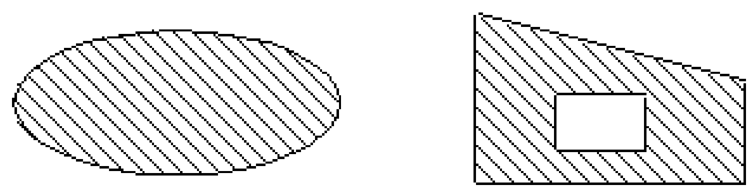
Multi-layer Perceptron (MLP)

- Multi-layer neural networks can solve problems that are not linearly separable
- Hidden units re-map input space into a space which can be linearly separated by output units.



Each hidden unit draws a line

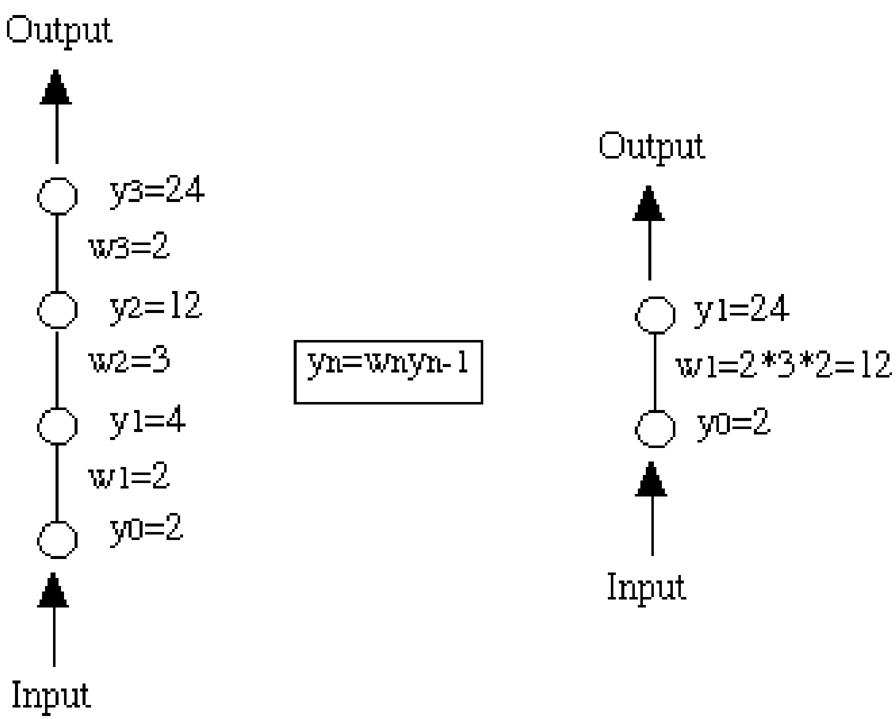




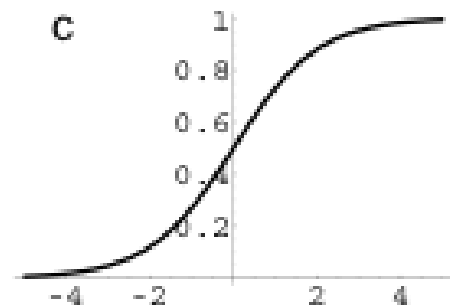
Output units “look” at regions (in/out)

Output Function in MLP

• Multi-layer networks should not use linear output functions because a linear transformation of a linear transformation remains a linear transformation.
• Therefore, such a network would be equivalent to a network with a single layer



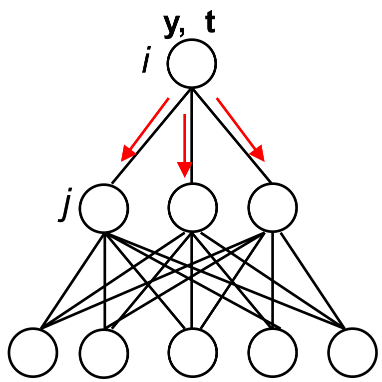
Sigmoid function is often used in MLP



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

Back-propagation of Error

In a simple perceptron, it is easy to change the weights so to minimize the error between output of the network and desired output.



$$\delta_i = t_i - y_i \qquad \Delta w_{ij} = \eta \delta_i x_j$$

$$\delta_i = \left(t_i - y_i\right) \dot{\Phi}\left(A_i\right)$$

in the case of non-linear output functions, add derivative of output

In an MLP, what is the error of the hidden units? This information is needed to change the weights between input units and hidden units.

The idea suggested by Rumelhart et al. in 1986 is to propagate the error of the output units backward to the hidden units through the connection weights:

$$\delta_j = \dot{\Phi}\left(A_j\right) \sum_i w_{ij} \delta_i$$

Once we have the error for the hidden units, we can change the lower layer of connection weights with the same formula used for the upper layer.

Algorithm

1. Initialize weights (random, around 0)

2. Present pattern $x_k^\mu = s_k^\mu$

QuickTime™ and a
TIFF (LZW) decompressor
are needed to see this picture.

3. Compute hidden $h_j^\mu = \Phi\left(\sum_k v_{jk} x_k^\mu\right)$

4. Compute output $y_i^\mu = \Phi\left(\sum_j w_{ij} h_j^\mu\right)$

5. Compute delta output $\delta_i^\mu = \Phi'\left(\sum_j w_{ij} h_j^\mu\right)\left(t_i^\mu - y_i^\mu\right)$ $\qquad \delta_i^\mu = y_i^\mu\left(1 - y_i^\mu\right)\left(t_i^\mu - y_i^\mu\right)$

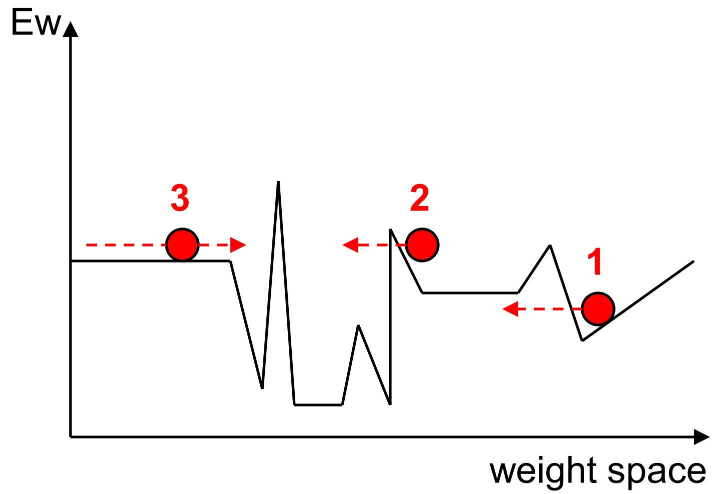
6. Compute delta hidden $\delta_j^\mu = h_j^\mu\left(1 - h_j^\mu\right)\sum_i w_{ij}\delta_i^\mu$

7. Compute weight change $\Delta w_{ij}^\mu = \delta_i^\mu h_j^\mu, \quad \Delta v_{jk}^\mu = \delta_j^\mu x_k^\mu$

8. Update weights $w_{ij}^t = w_{ij}^{t-1} + \eta \Delta w_{ij}^\mu, \quad v_{jk}^t = v_{jk}^{t-1} + \eta \Delta v_{jk}^\mu$

Using Back-Propagation

Error space can be very hard to explore: local minima and flat areas



1. Large learning rate: take large steps in the direction of the gradient descent

2. Momentum: add direction component from last update $\Delta w_{ij}^{t} = \eta \delta_i + \alpha \Delta w_{ij}^{t-1}$

3. Additive constant: keep moving when no gradient $\delta_i^{\mu} = \left(\Phi' + k\right)\left(t_i^{\mu} - y_i^{\mu}\right)$

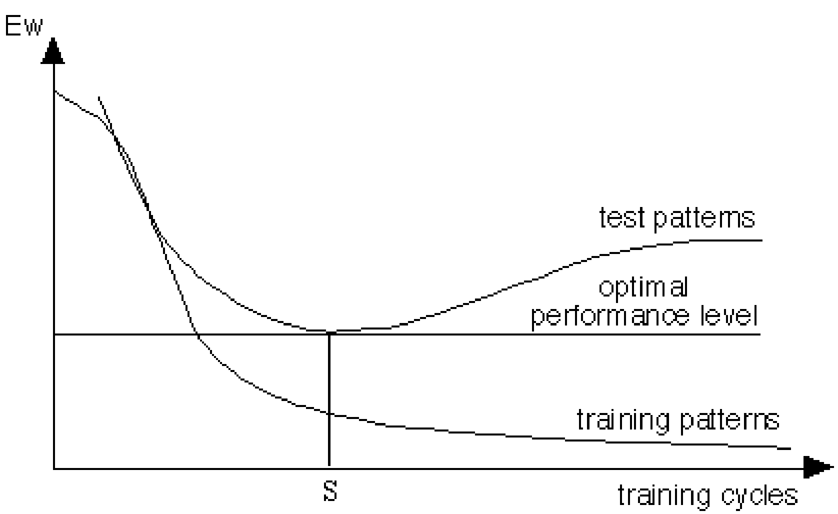
Preventing Over-fitting

QuickTime™ and a
TIFF (LZW) decompressor
are needed to see this picture.

Ideally, one wants that the network generalizes to new data.

Too many weights may lead to overfitting of training data.

Not easy to tell appropriate network architecture.

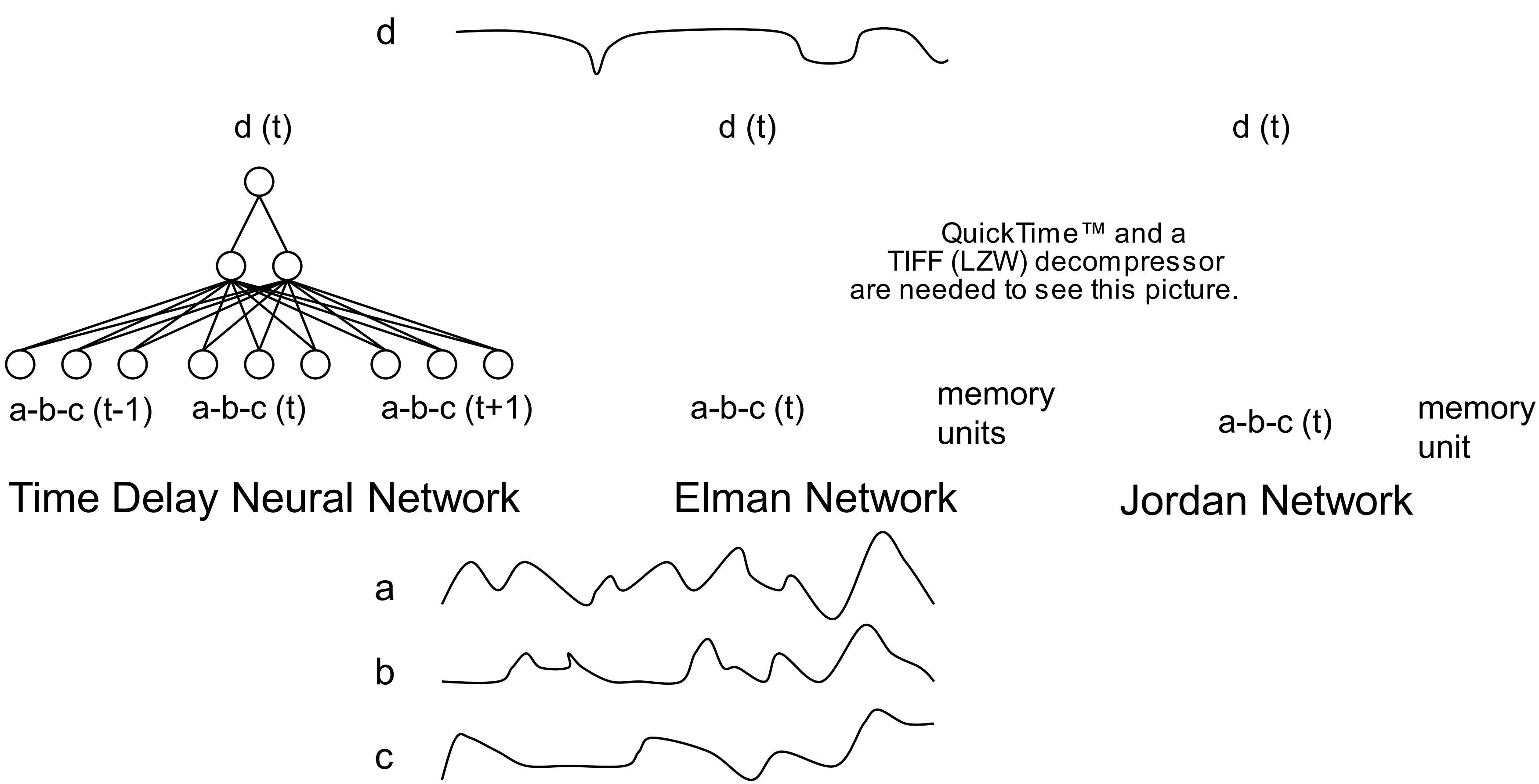


Solution: Careful training

Divide available data into:
- training set (for weight update)
- testing set (for error monitoring)
Stop training when error for test set grows

Time Series

Extraction of time-dependent features is necessary for time-series analysis



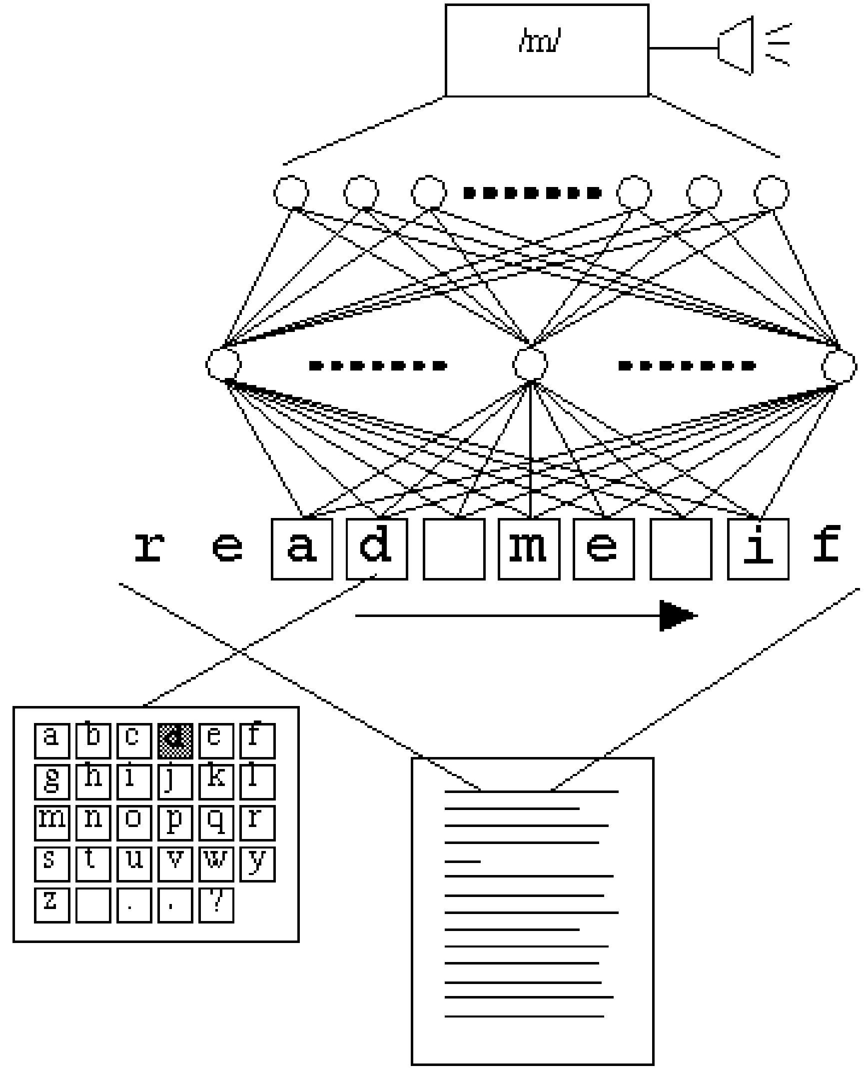
NETtalk

A neural network that learns to read aloud written text:

- 7 x 29 input units encode characters within a 7-position window(TDNN)
- 26 output units encode english phonemes
- approx. 80 hidden units

Training on 1000-word text, reads any text with 95% accuracy

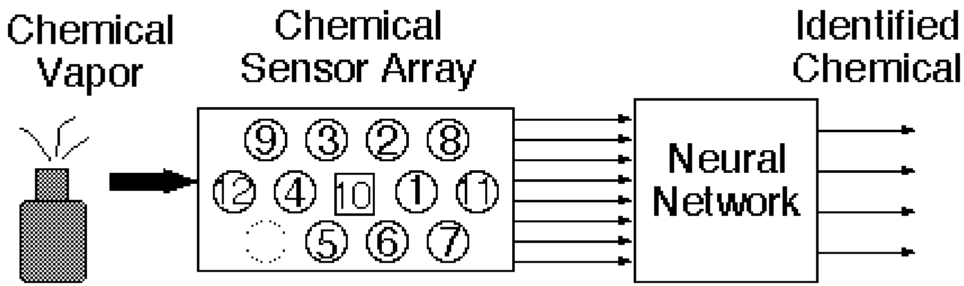
Learns like humans: segmentation, bla-bla, short words, long words



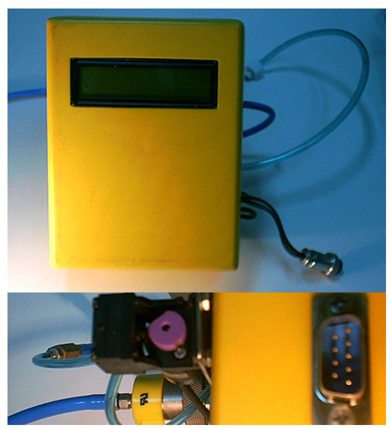
[Sejnowski & Rosenberg, 1987]

Artificial Nose

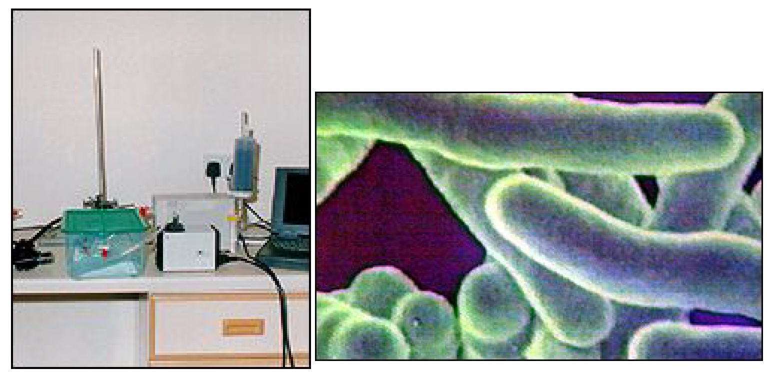
The human brain recognizes millions of smell types by combining responses of only 10,000 receptors. Smell detection is a multi-billion industry (food, cosmetics, medicine, environment monitoring...). Human detection: costly, fatigue, history, aging, subjective.



landmine detection
Tufts University

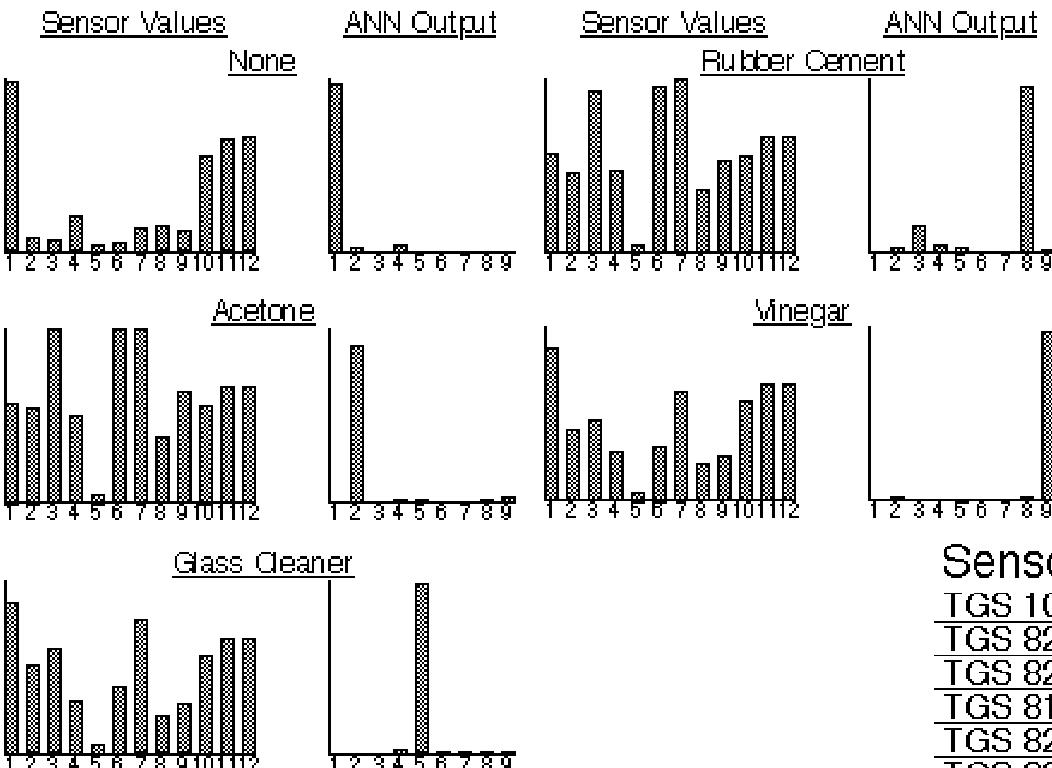


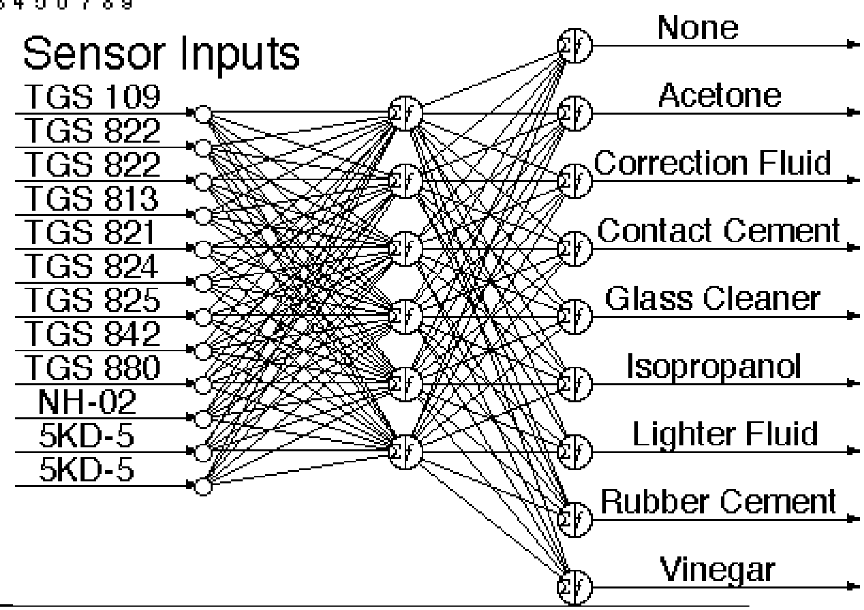
food quality
Pampa Inc.



tubercolosis diagnosis
Cranfield University

Neural Net Recognition

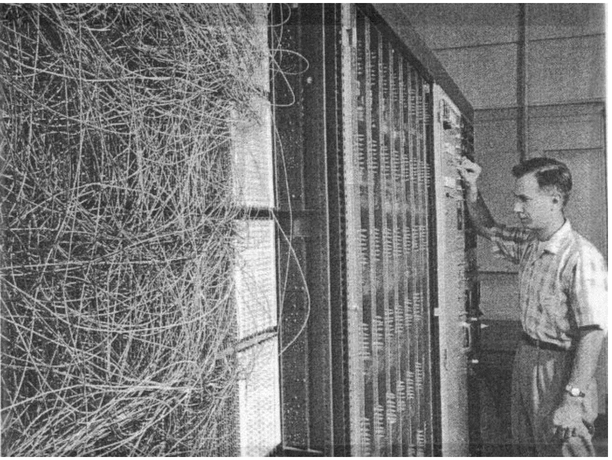




[Keller et al., 1994]

Neural Hardware Implementations

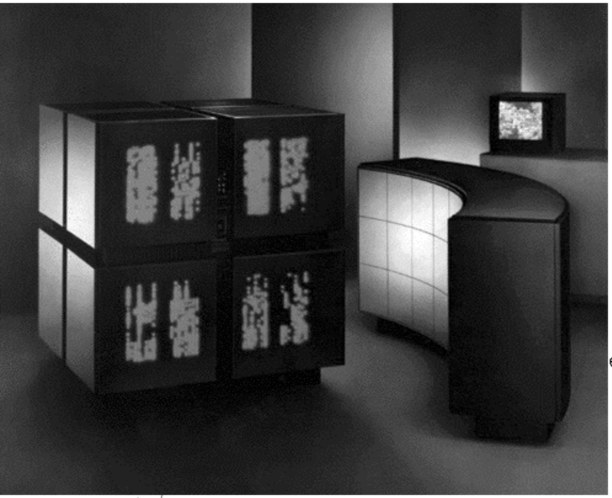
Mark I Perceptron (1960)



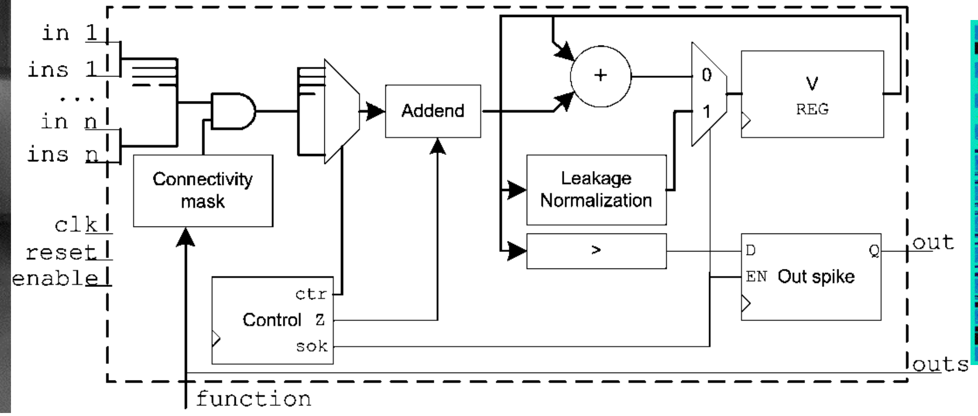
QuickTime™ and a
TIFF (Uncompressed) decompressor
are needed to see this picture.

Optical (1990)

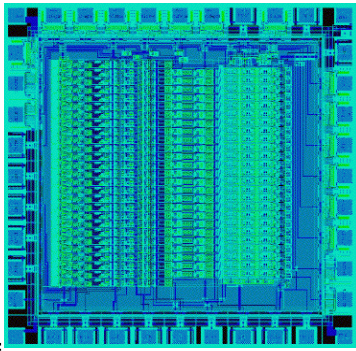
Connection Machine (1990)



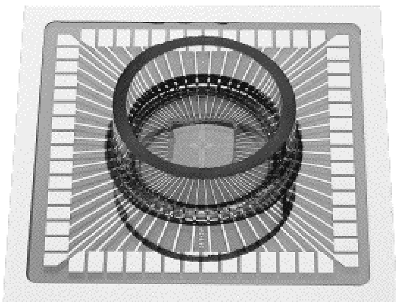
FPGA (2000)

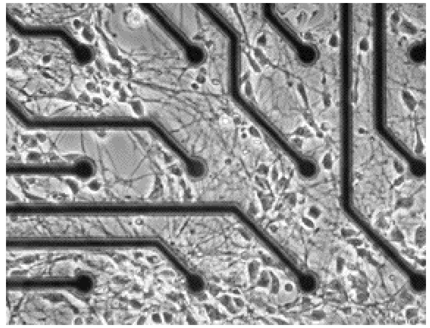


aVLSI (2000)



Hybrid Neural Systems: *Multi-Electrode-Array*

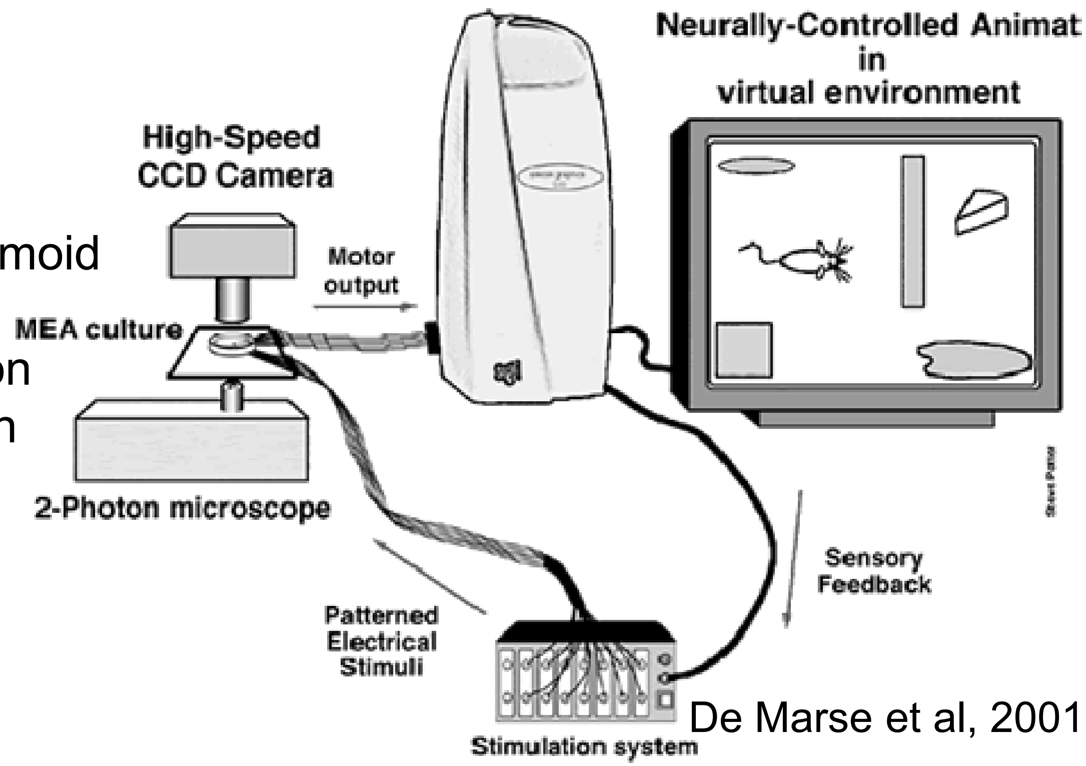




<u>Records/stimulates groups of neurons</u>

Neurons in sealed container (only oxigen and carbon dioxide exchange)
Activity for several months

60 electrodes spaced every 200 μm
Spike count over 200 ms through sigmoid
Cluster patterns of 60 values
Associate each cluster with one action
Agent’s sensors to neuron stimulation



De Marse et al, 2001

Hybrid Neural Systems: *Field-Effect-Transistor*

<u>Records/stimulates single neuron</u>

Monitor biological neural communication
Connect distant neurons by electrical connections
Stimulate neurons and record network activity
Grow biological networks
Interface with artificial networks

neuron
stabilizers

protein layer
silicon-dioxide layer

QuickTime™ and a
TIFF (LZW) decompressor
are needed to see this picture.

source-drain current

Fromherz, 2003