Bio-Inspired Artificial Intelligence

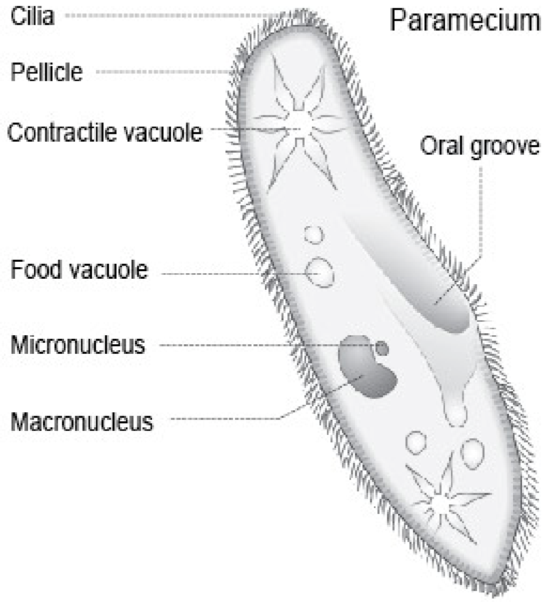
Lecture 3: Spiking Neural Networks

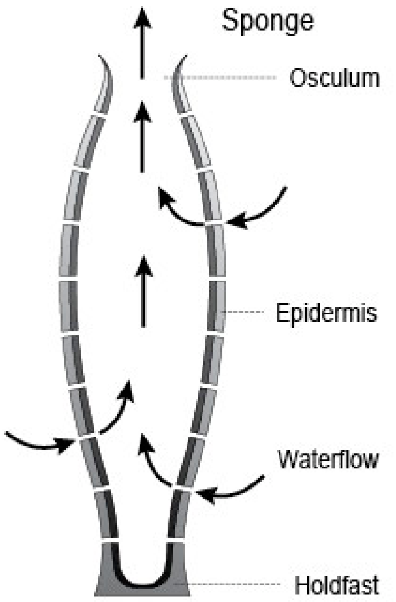
http://www.informatik.uni-hamburg.de/WTM/

Spiking Neural Networks: Motivation

Why Nervous Systems?

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



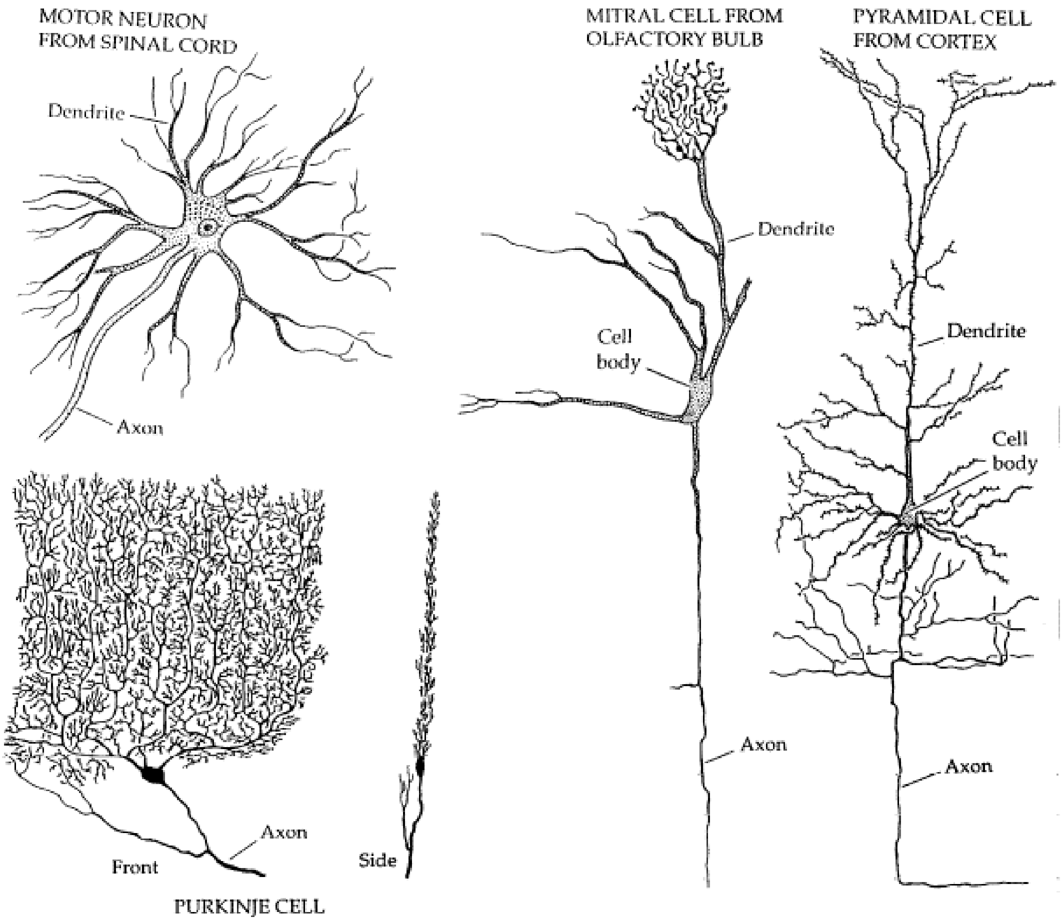


What advantages do nervous systems provide?

1) Selective transmissions of signals across distant areas=more complex bodies
2) Complex adaptation to environmental changes

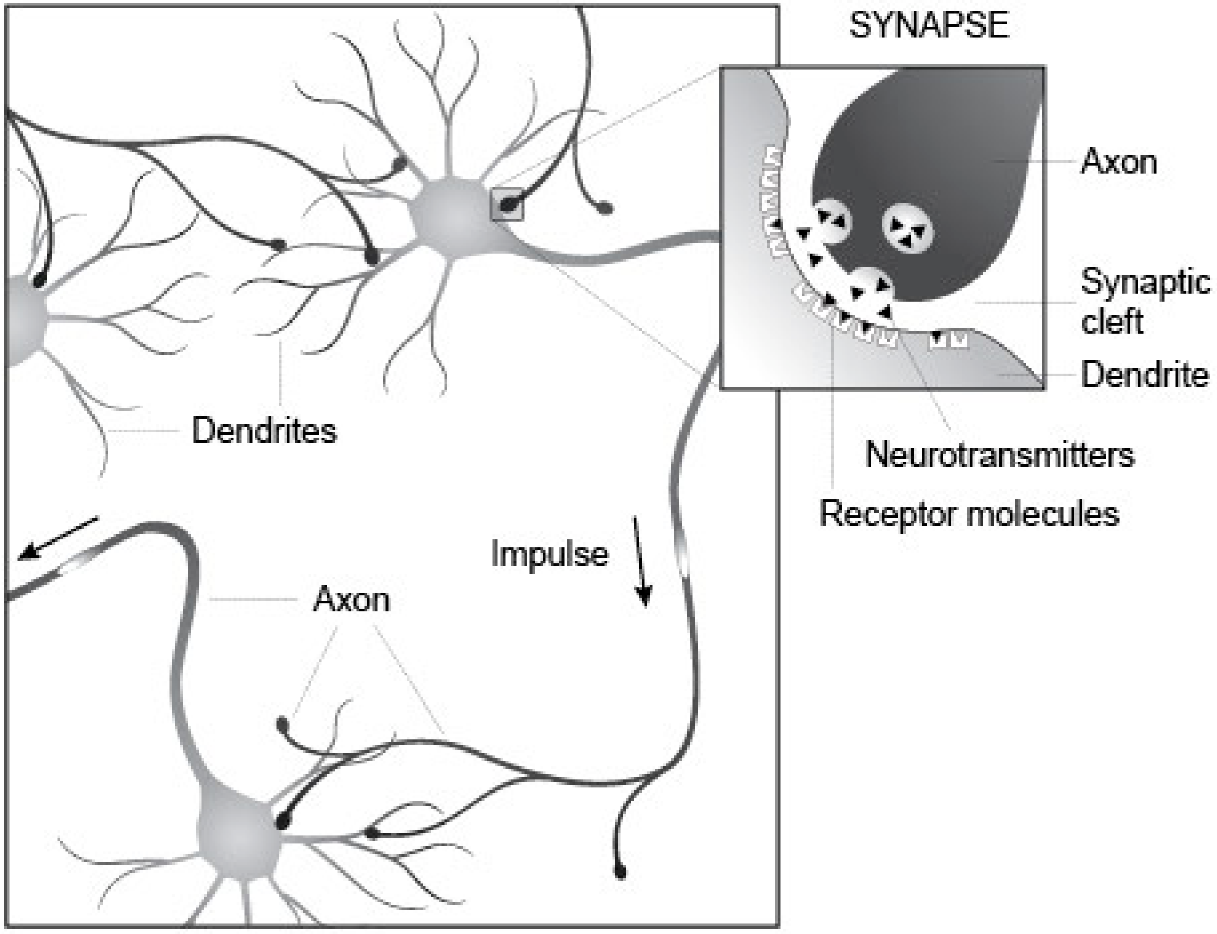
What makes brains different?

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

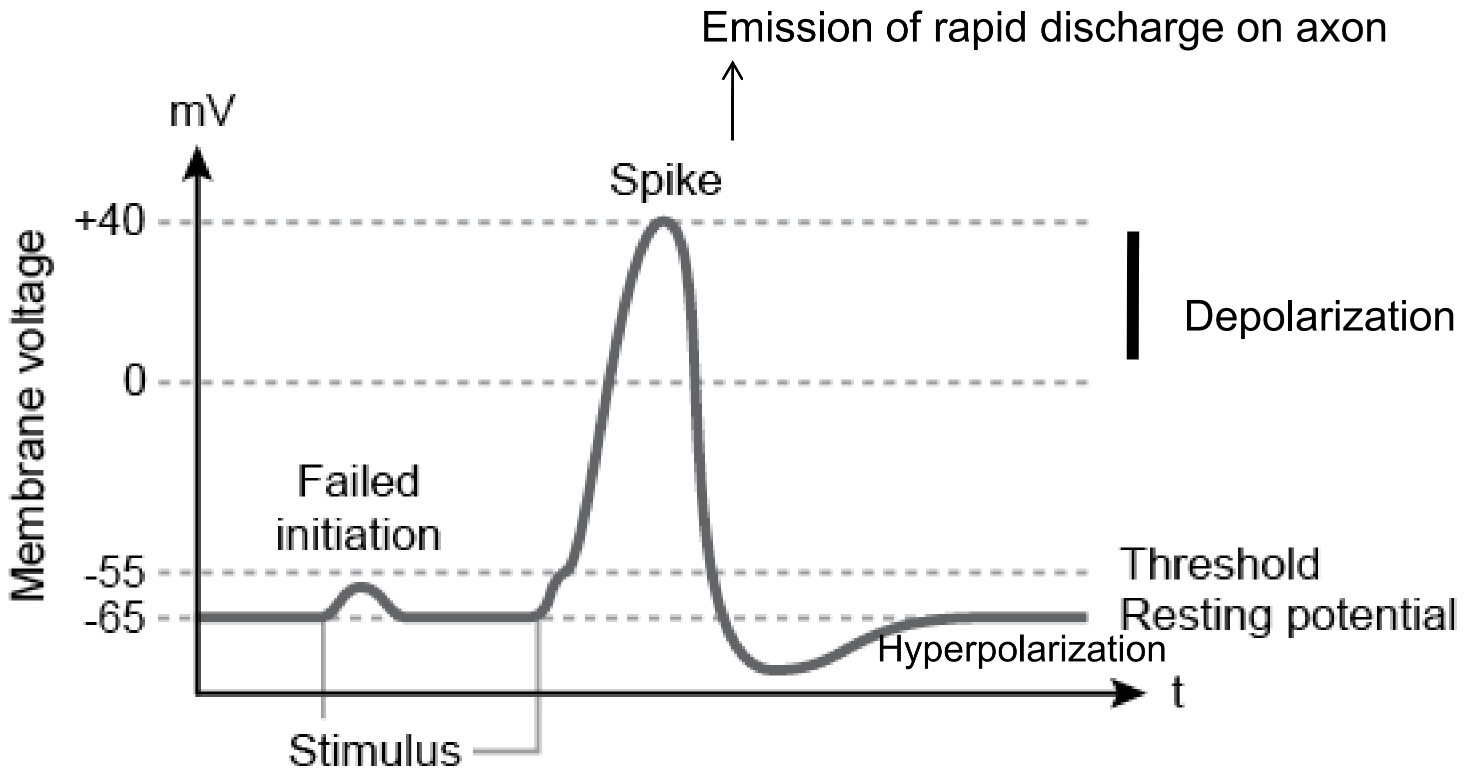


- Evolution of the brain seems to occur mainly in the architecture, that is how neurons are interconnected.
- First classification of neurons by Cajal in 1911 was made according to their connectivity patterns

Biological Neurons

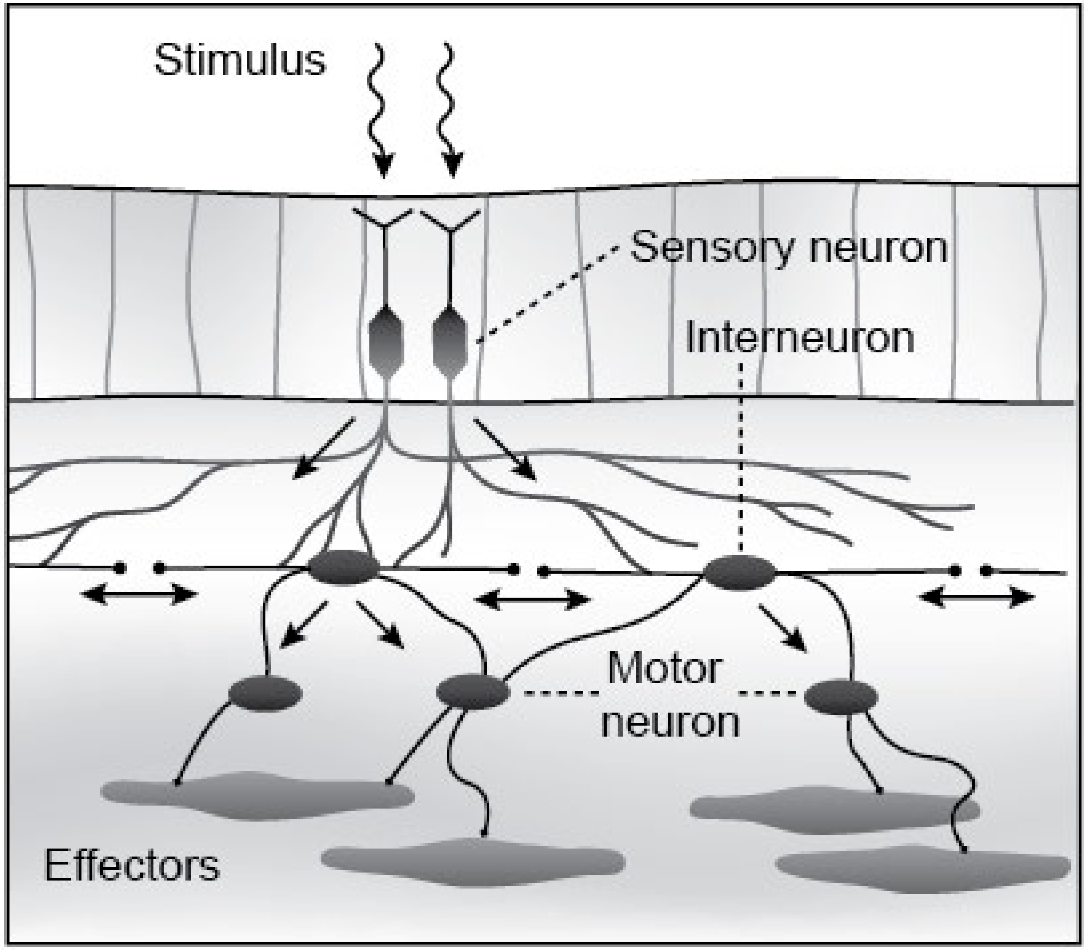


Membrane Dynamics



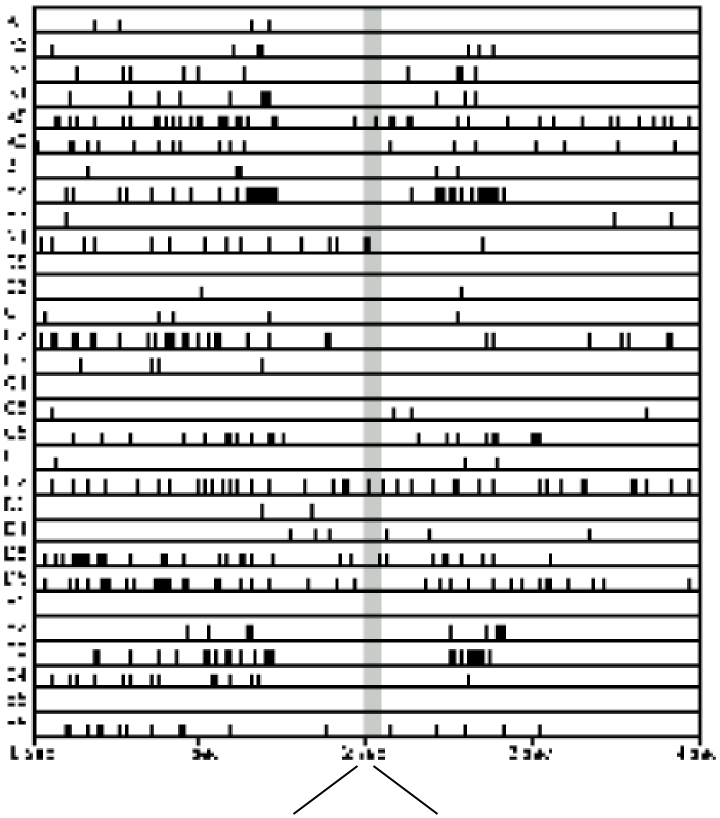
- This cycle lasts approximately 3-50 ms, depending on type of ion channels involved (Hodgkin and Huxley, 1952)

Types of Neurons



Interneurons can be
1- Excitatory
2- Inhibitory

How do Neurons Communicate?



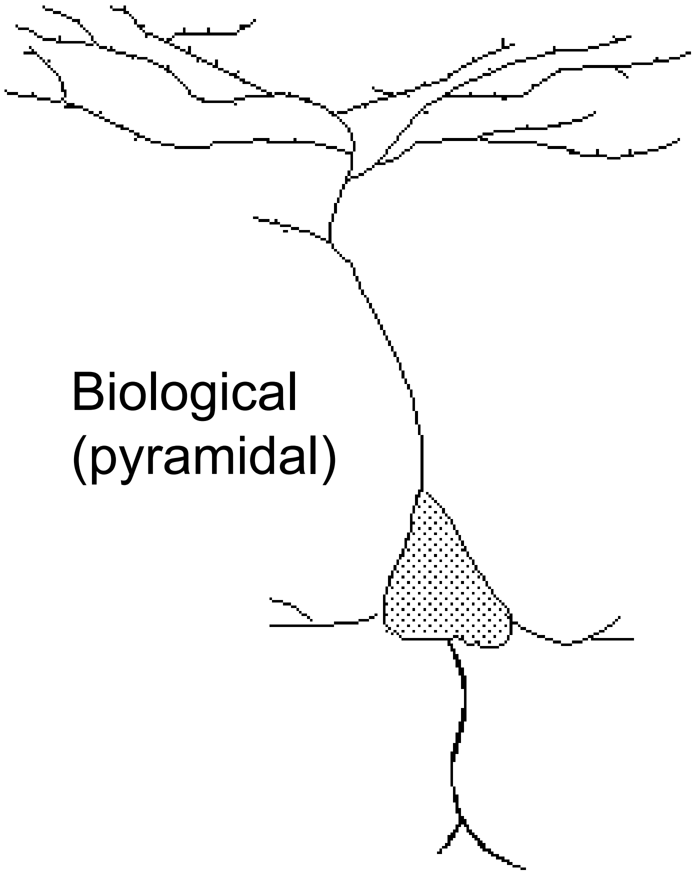
100 ms

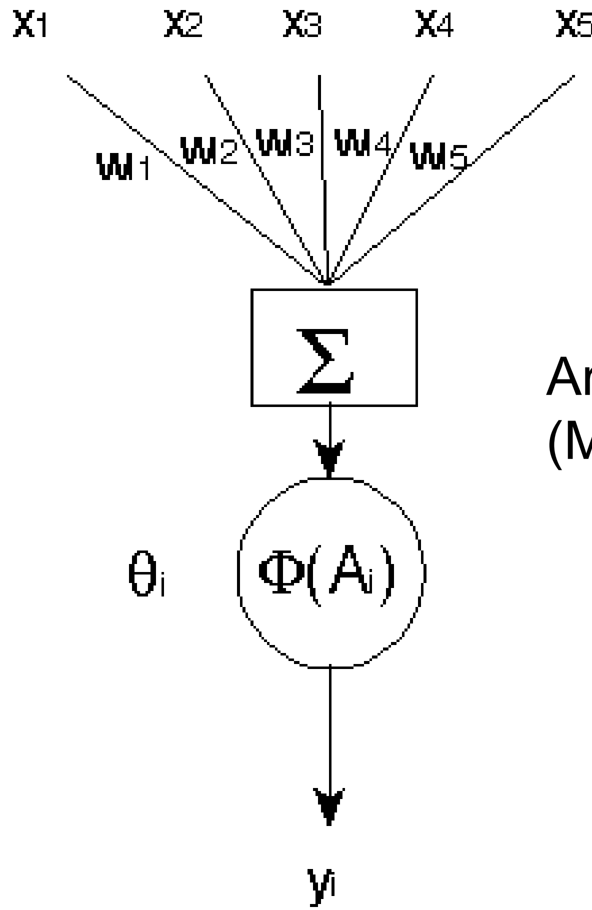
Firing rate → McCulloch-Pitts → Connectionism

Firing time → Spiking neurons → Computational Biology

Biological and Artificial Neuron

Biological
(pyramidal)





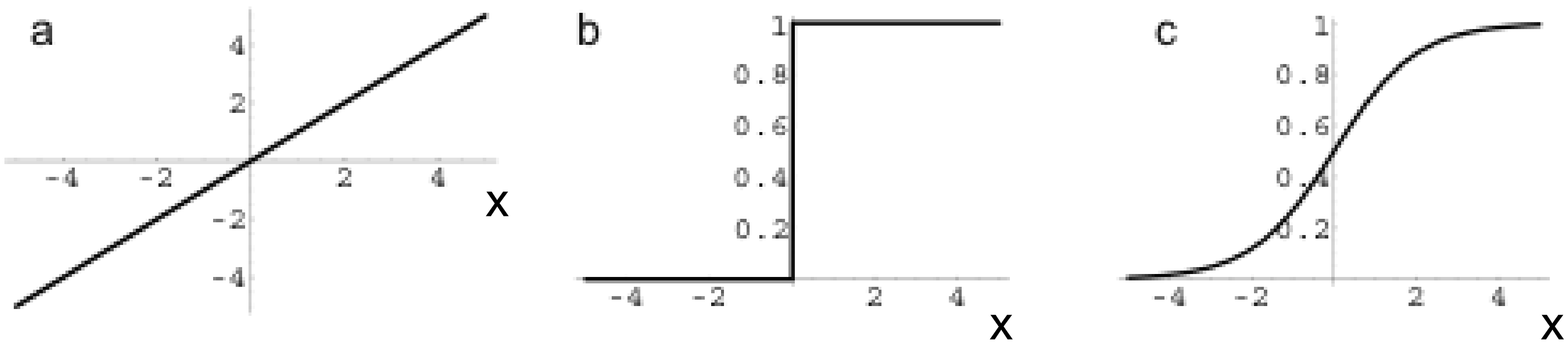
Artificial
(McCulloch-Pitts)

direction of signal transmission

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

Different Forms of Activation Functions

Identity	Step	Sigmoid
$\Phi(Ai)$	$\Phi(Ai)$	$\Phi(Ai)$



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

$$\Phi(Ai) = \frac{1}{1 + e^{-kAi}}$$

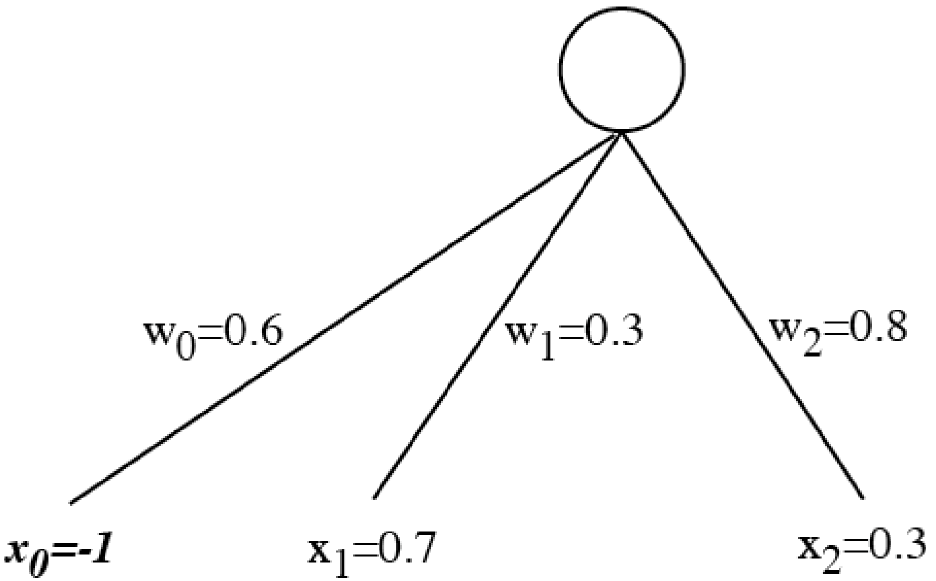
$$\Phi(A_i) = \tanh(kA_i)$$

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*** -1.0

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

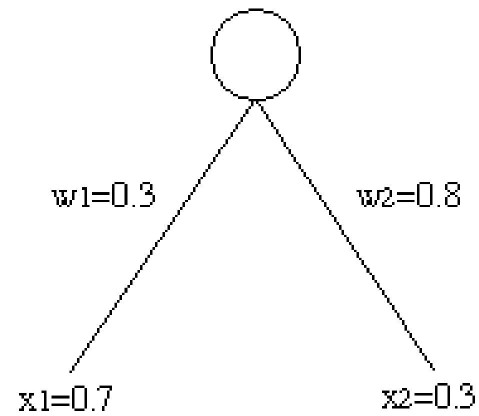
$$y_i = \Phi(A_i) = \Phi\left(\sum_{j=0}^{N} w_{ij} x_j\right)$$



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

Signalling Similarity

The output of a neuron can be 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. Similarity/Angle between two vectors is:

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

where the vector length is:

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

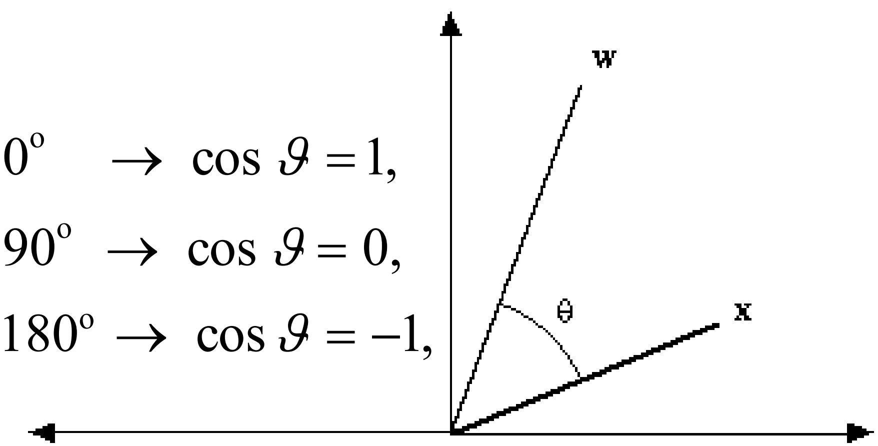
3. Output signals input familiarity

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

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

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

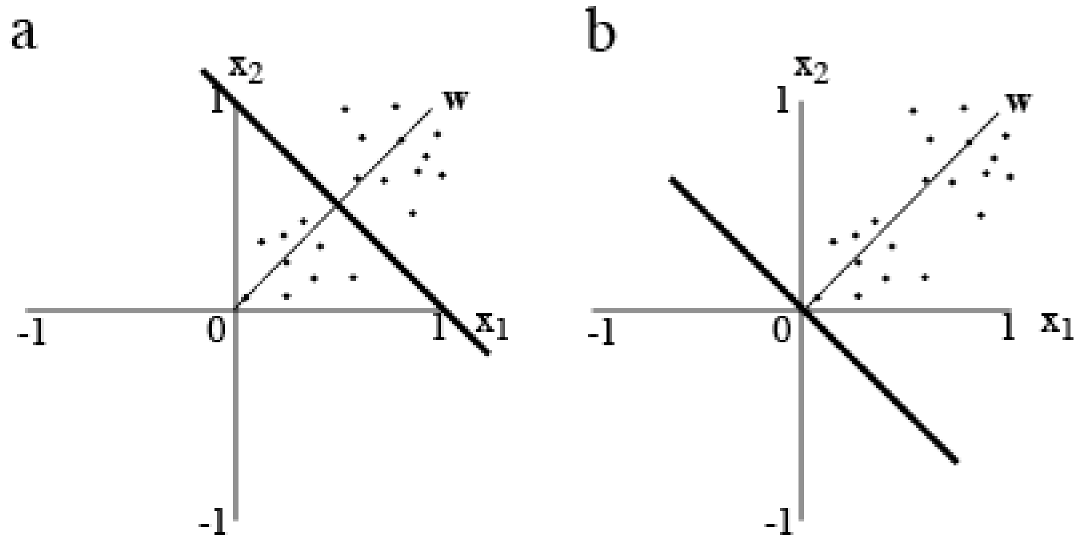
$$\vartheta = 180^{\circ} \rightarrow \cos \vartheta = -1,$$



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:

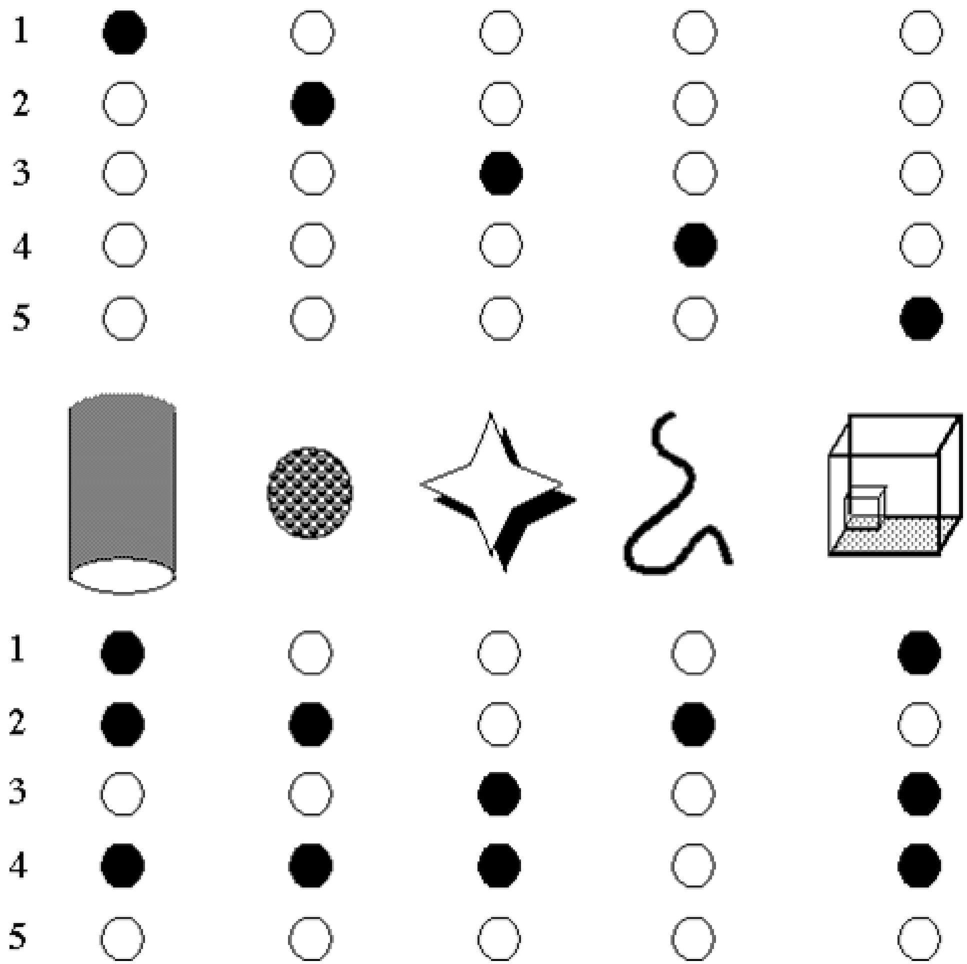
$$w_1 x_1 + w_2 x_2 - \vartheta = 0 \qquad x_2 = \frac{\vartheta}{w_2} - \frac{w_1}{w_2} x_1$$



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

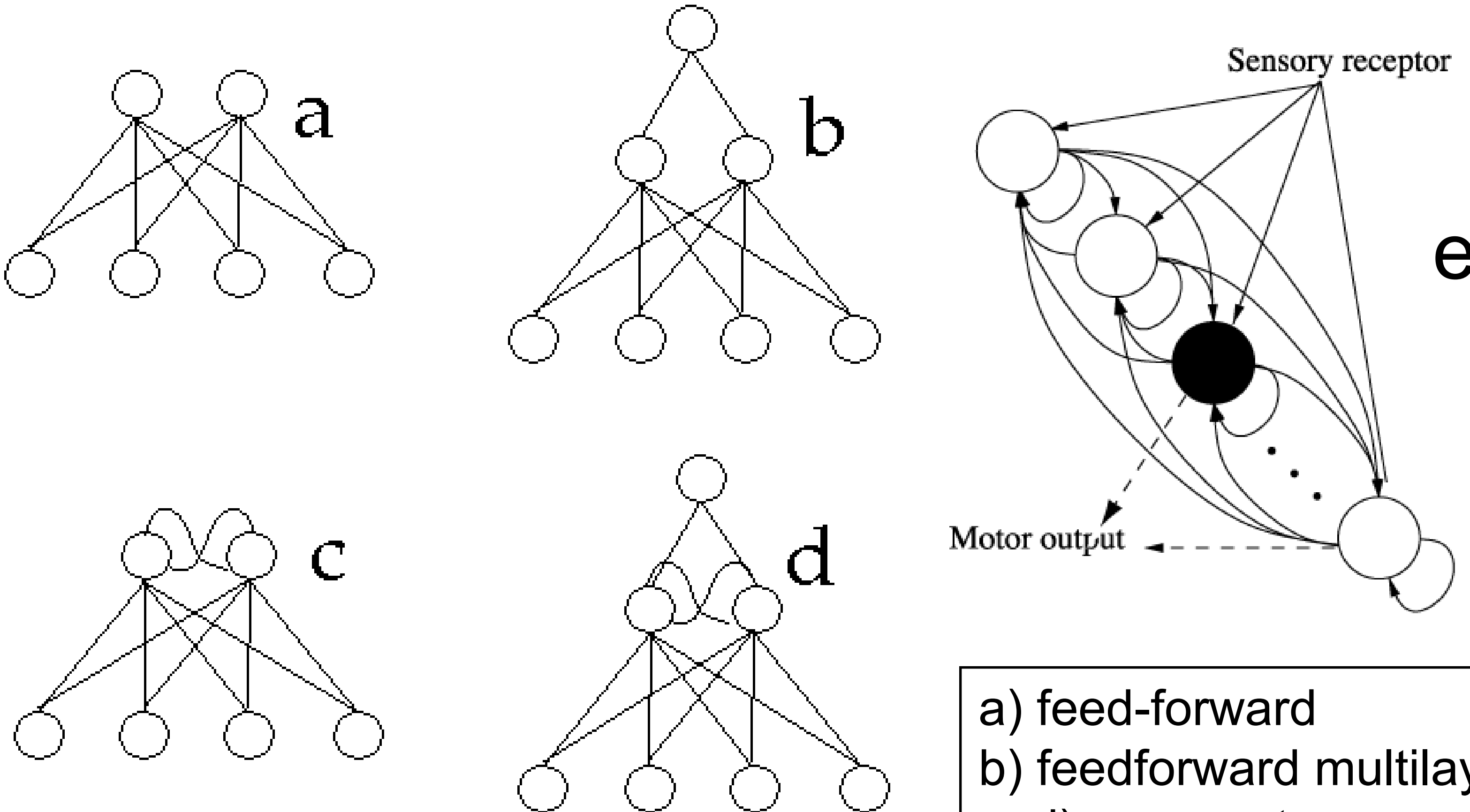
thin line is the weight vector, thick line is separation line

Overview on 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

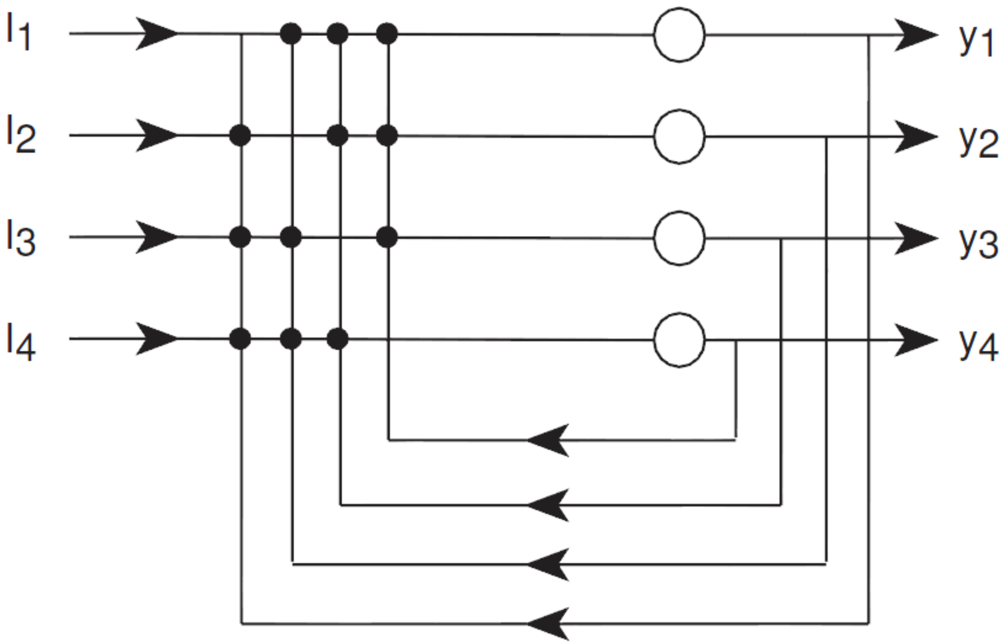
Example Architectures



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

Example: Auto-Associative Network

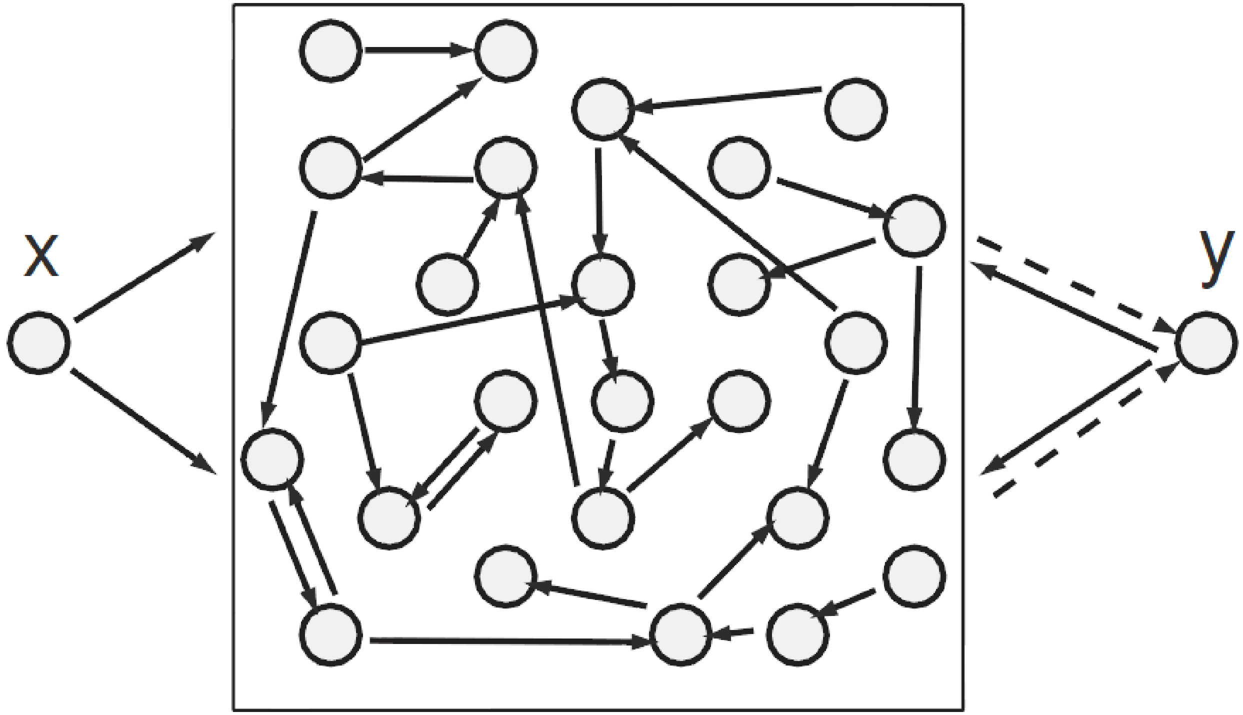
- Fully connected single-layer network (no self-reference)
- Memorizes and reconstructs pattern
- Input I can be incomplete or corrupted
- Iterative computation of output y_i



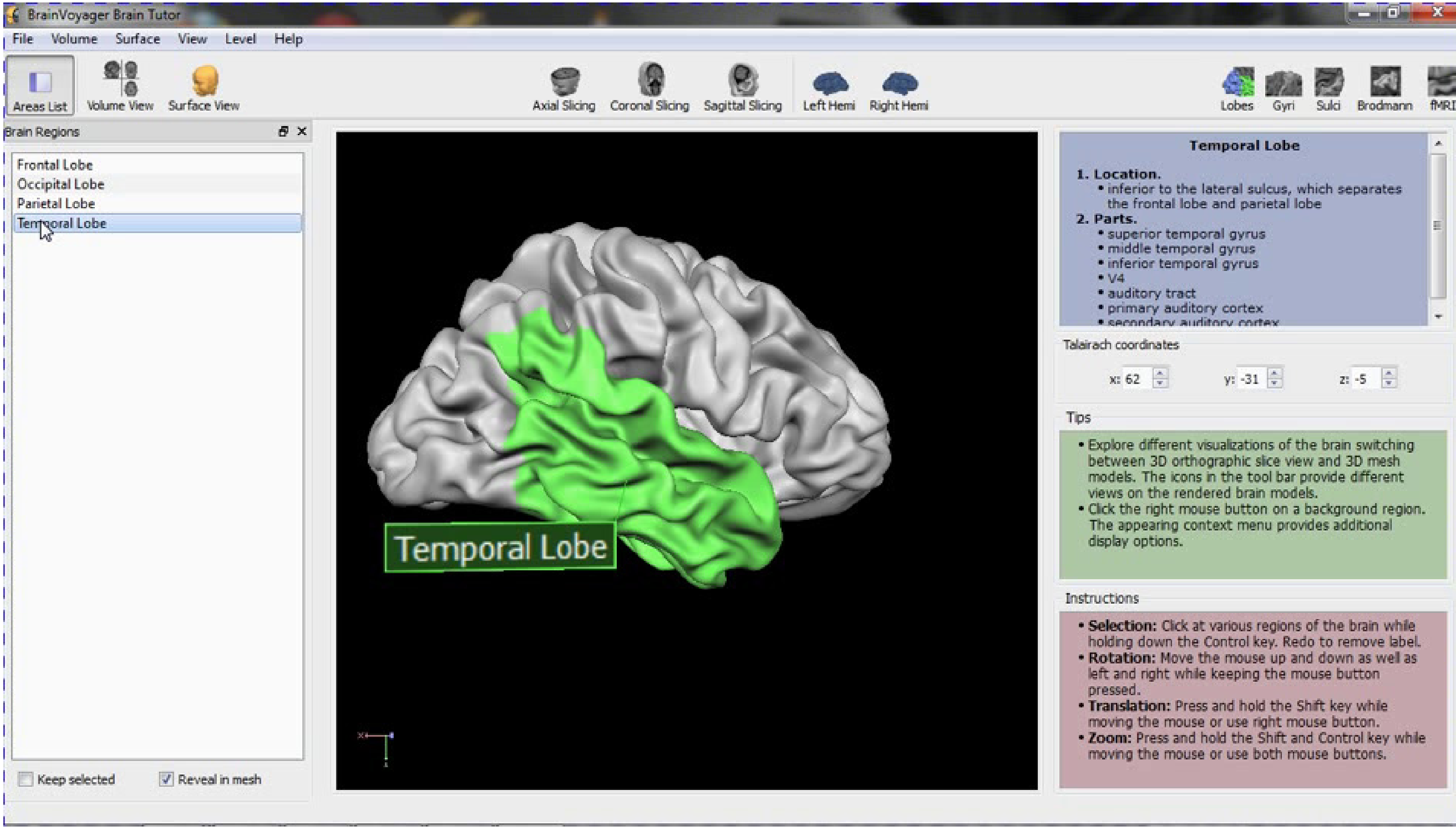
$$y_i = \Phi\left(I_i + \sum_{j \neq i}^{N} \omega_i x_i\right)$$

Example: Echo State Network

- Random interconnected hidden neurons (typically 50-1000)
- Input layer and bidirectional output layer
- Hidden Neurons loosely coupled (with a probability)
- Sub-networks as echo functions between input and output



Brain Voyager Brain Tutor Demo

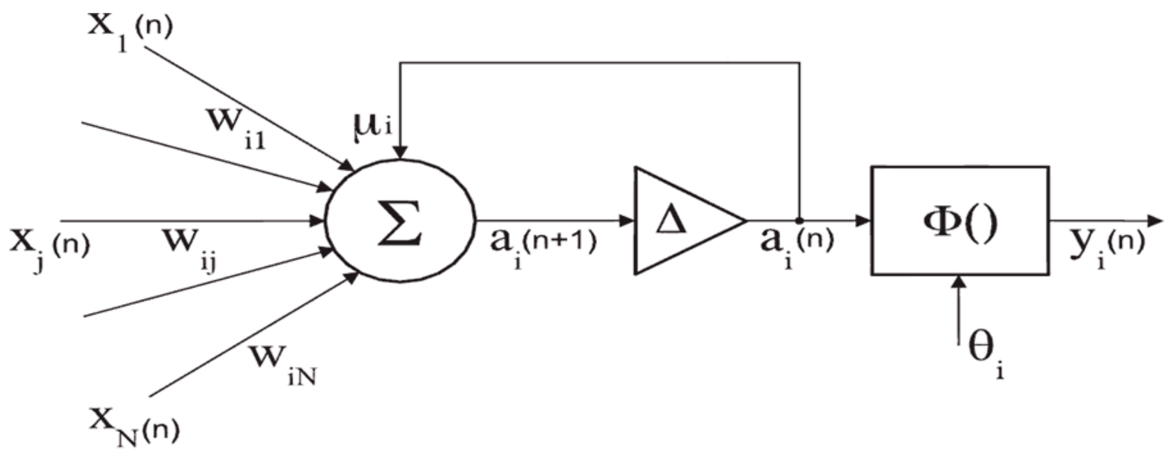


http://www.brainvoyager.com/BrainTutor.html

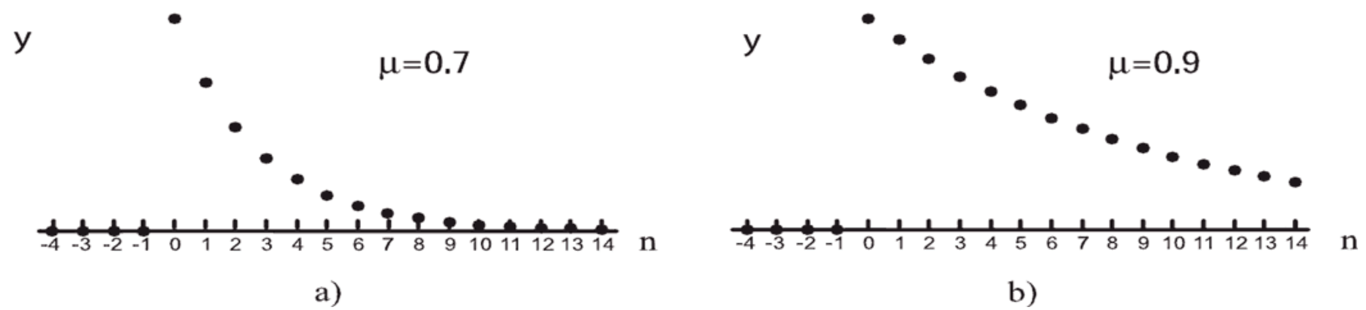
Towards Dynamic Neuron Models

- Model presented so far is rather static
- Output is determined by current input
- No temporal sequencing
- In some task: Time matters (e.g. speech perception)
- Additional feedback loop between input- and output neuron
- Provides temporal delay

Discrete Dynamic Neuron Model



- Δ: Time delay between current activation $\alpha_i(n)$ and update step $\alpha_i(n+1)$
- Delay keeps the activation α_i for an amount of Δ
- μ_i: recurrent connection weight



Activation decay for different μ_i

Discrete Dynamic Neuron Model

- Computation of update of the discrete model:

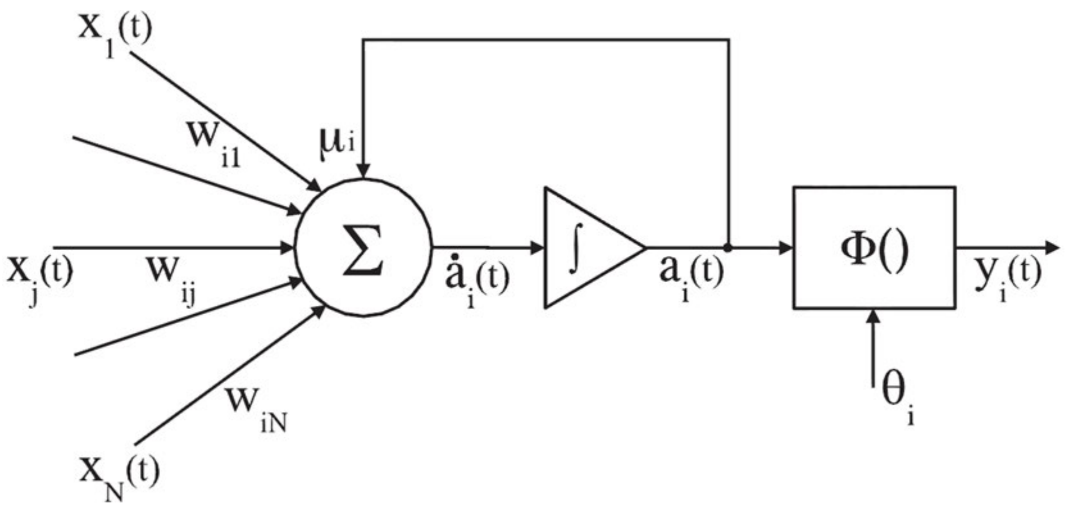
$$a_i(n+1) = \mu_i a_i(n) + \sum_{j=1}^{N} \omega_{ij} x_j(n)$$

- Computation of output signal:

$$y_i(n) = \Phi(a_i(n) - \theta_i)$$

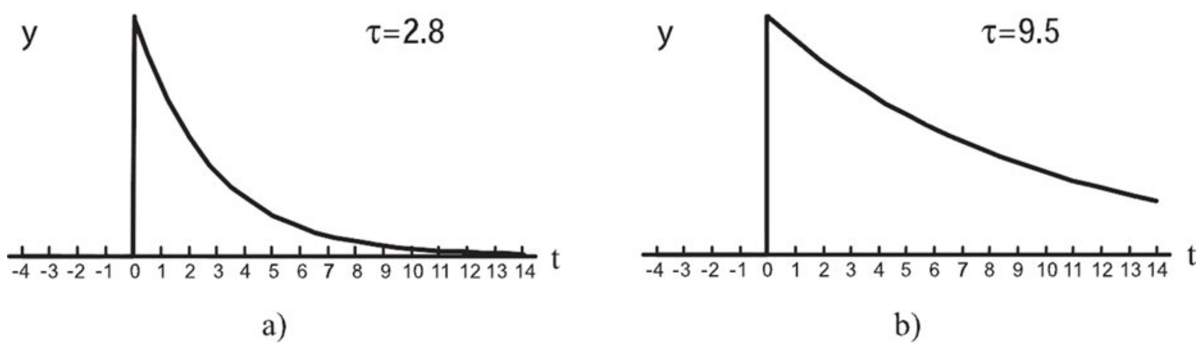
- So far for the discrete case: but how to model a continuous neuron model?

Continuous Neuron Model



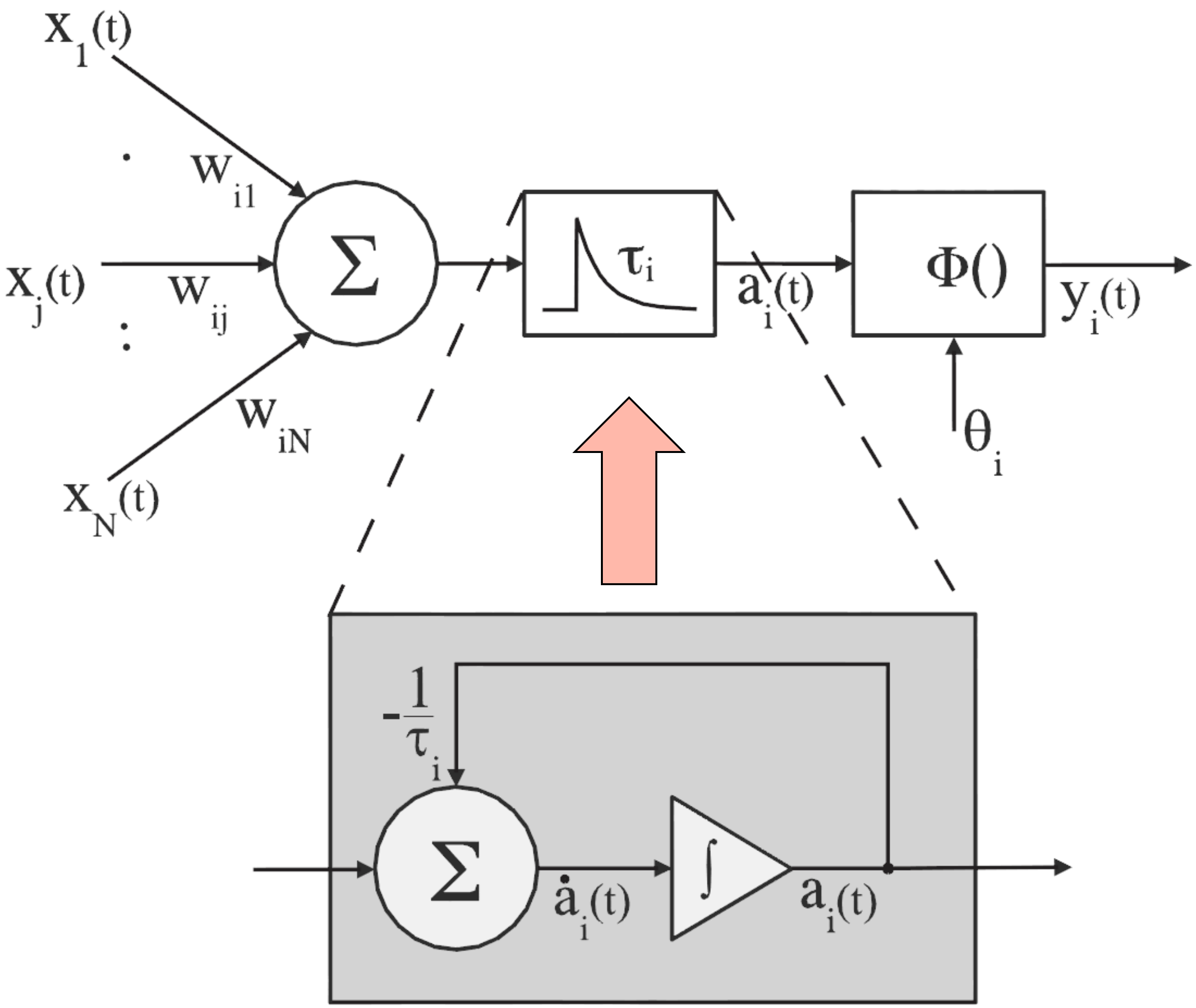
- Time delay: from Δ to $\int$
- Update is derivative of activation over time t

$$e^{-t/\tau_i}$$



μ takes the form of exponential decay e^{-t/τ_i} for $\tau > 0$

Compact Representation



- Note: μ (discrete case) now becomes τ (continuous case)

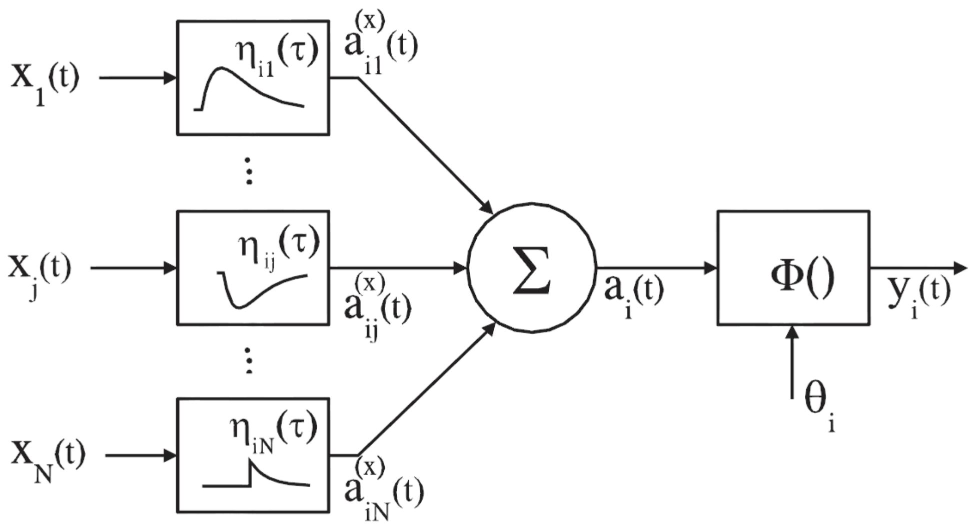
Approximation of Continuous Model

- Approximation of activation update
- Derivation of activation over time t:

$$\frac{da_i(t)}{dt} = -\frac{1}{\tau_i} a_i(t) + \sum_{j=1}^{N} \omega_{ij} x_j(t)$$

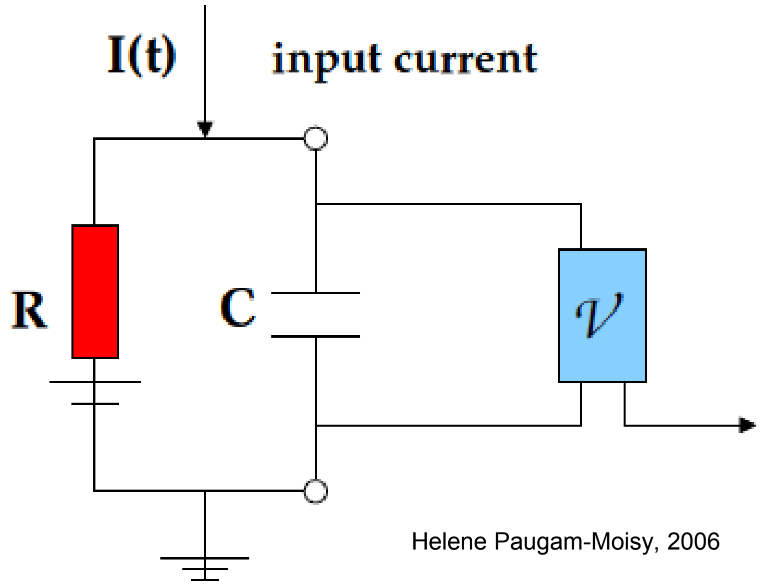
- Displays exponential dynamics of activation at neurons membrane
- ***Leaky Integrator***: RC-circuit equivalent

Generalized Model



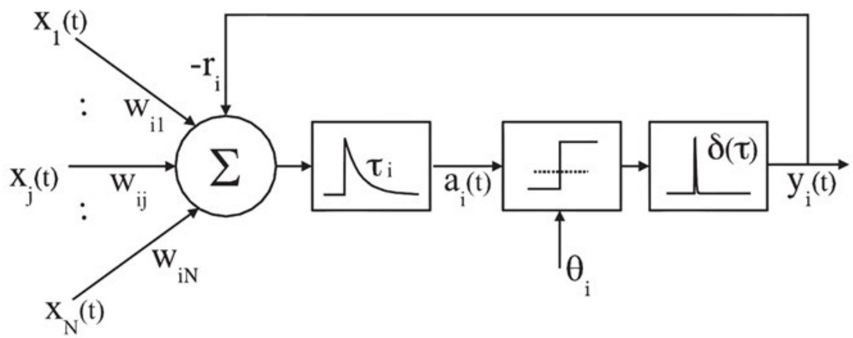
- Different temporal dynamics for each input
- Generalizes more to biological case: each synapse contributes with different temporal characteristics to neuronal activation
- η_{ij}: synaptic kernel
- Temporal convolution of input x_i with kernel

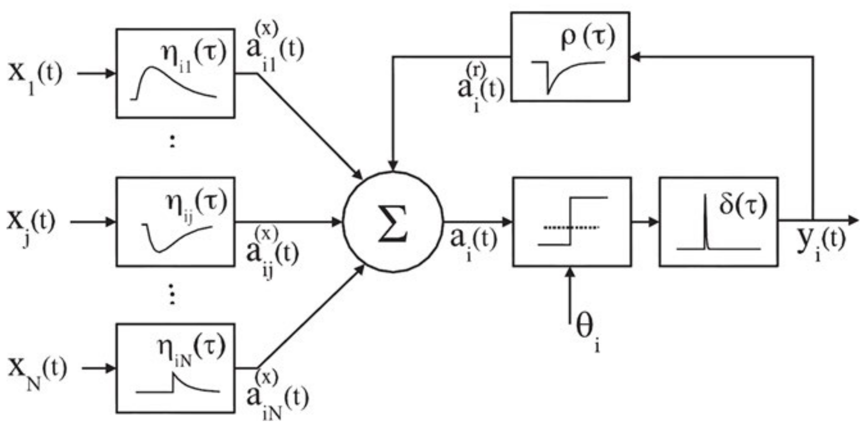
(Leaky) Integrate and Fire Model



- Simplification of Hodgkin and Huxley model
- Describes voltage difference of a capacitor: equations just of first-order linear differential form
- Time constants τ approximated by resistors

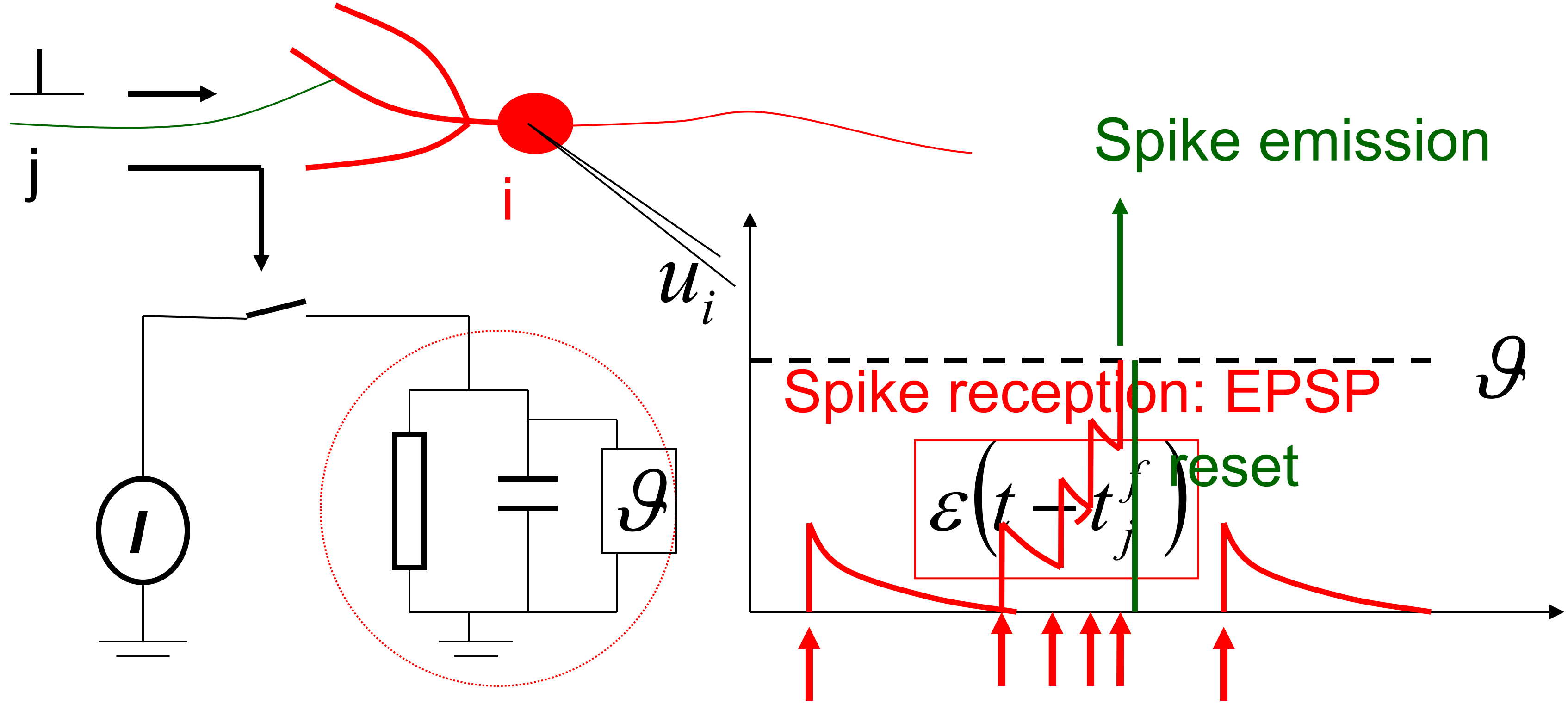
Integrate and Fire Model





- Integrate and Fire model
  - Spikes as function of time: $\delta(\tau)$ is Dirac-function, representing pulses/spikes (pulse generator)
  - Strong negative feedback: -r models refractory period after spike emission
- Spike Response Model (more sophisticated input – Gerstner99)
  - Negative feedback modeled by exponential function
  - Describes process for neuron to smoothly get into resting state (refractioness)

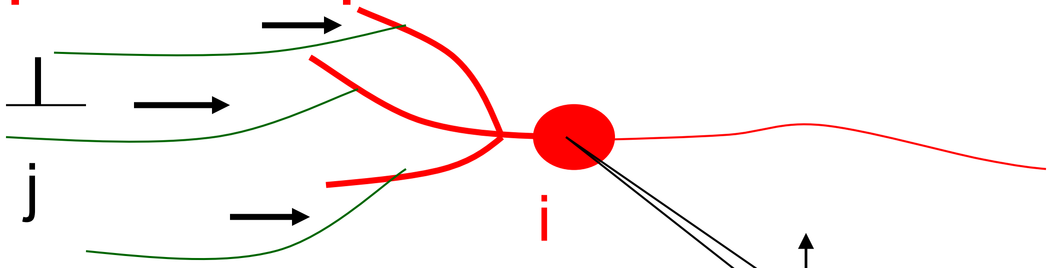
Integrate-and-fire Model details (Gerstner, Kistler)



$$\tau \cdot \frac{d}{dt} u_i = -u_i + RI(t)$$ linear

$$u_i(t) = \vartheta \Rightarrow$$ Fire+reset threshold

Spike Response Model details (Gerstner, Kistler)

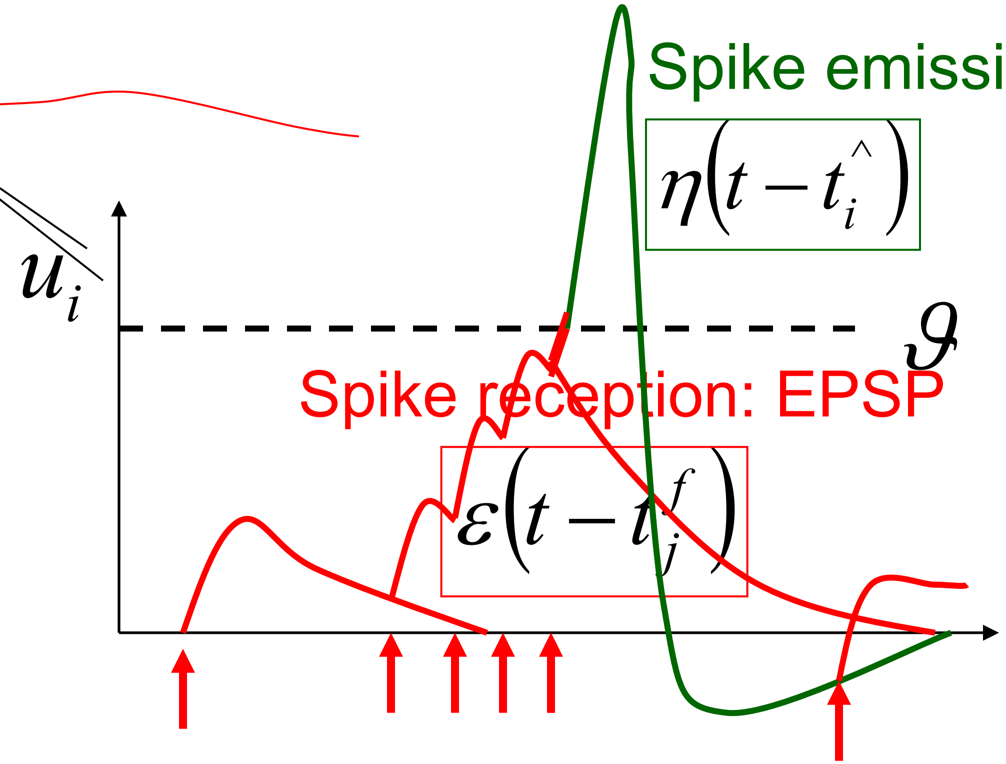


Spike reception: EPSP

$$\boxed{\varepsilon\left(t - t_j^f\right)}$$

Spike emission: AP

$$\boxed{\eta\left(t - \hat{t}_i\right)}$$



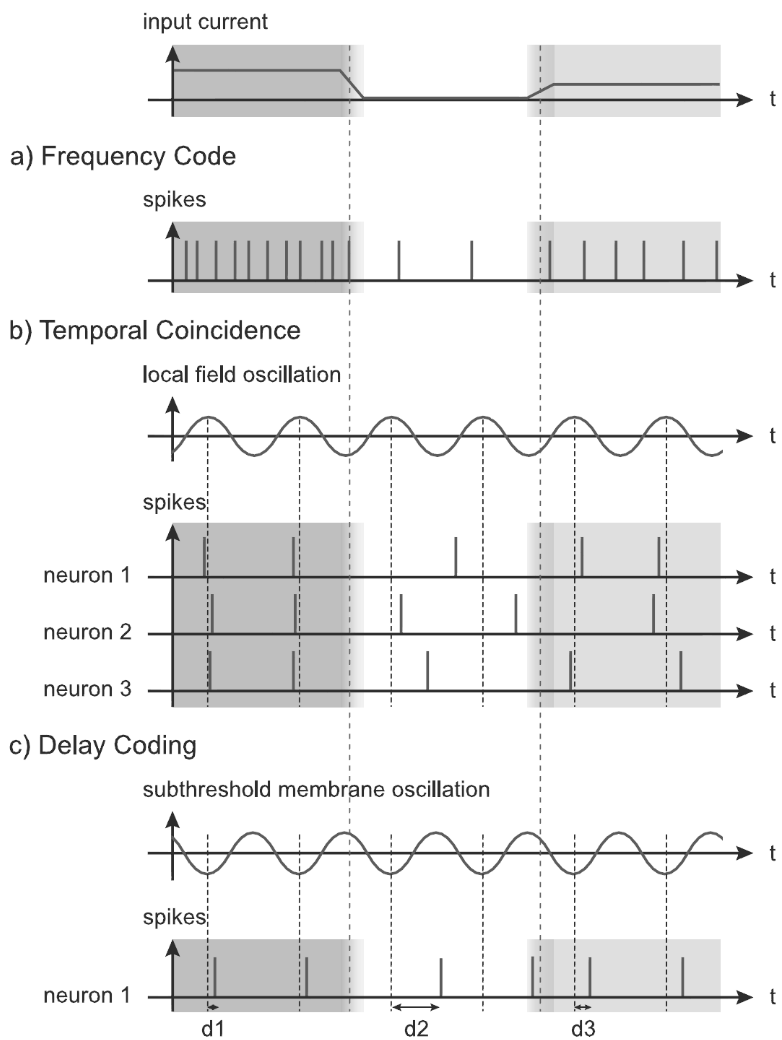
Last spike of i

All spikes, all neurons

$$u_i(t) = \underbrace{\eta\left(t - \hat{t}_i\right)} + \sum_j \sum_f w_{ij}\, \underbrace{\varepsilon\left(t - t_j^f\right)}$$ linear

$$u_i(t) = \vartheta \Rightarrow \text{Firing:} \quad \hat{t}_i = t$$ threshold

Spike Encoding



- Spike represents presence or absence of stimulus: binary decision
  - a) Firing rate represents strength of stimulus (Frequency code hypothesis)
  - b) Number of neurons firing coincidentally represent stimulus intensity (Temporal coincidence hypothesis)
  - c) Amount of spike response delay according to baseline signal determines stimulus: less delay when strong stimulus and vice versa (Delay coding hypothesis)

Synaptic Plasticity (1)

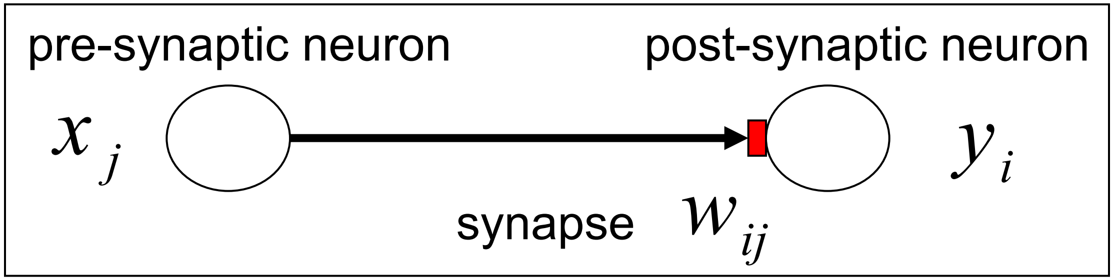
- Hebb rule (1949):

"Cells that fire together, wire together"

- i.e.: activation correlations strengthen synaptic connections
- Connections constitute the brain plasticity
- Plasticity is measure for complex task solving capabilities (compare to low-level organisms)
- Hebbian Learning: Associative Learning

Synaptic Plasticity (2)

- Learning is experience-dependent modification of connection weights



Hebb rule formalized: $$\Delta w_{ij} = x_j y_i$$

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

[0,1]

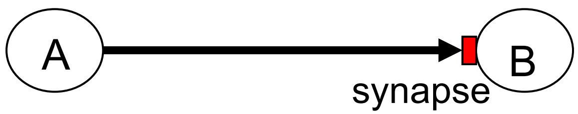
Hebb's rule suffers from ***self-amplification*** (unbounded growth of weights)

How do Neurons Learn? (1)

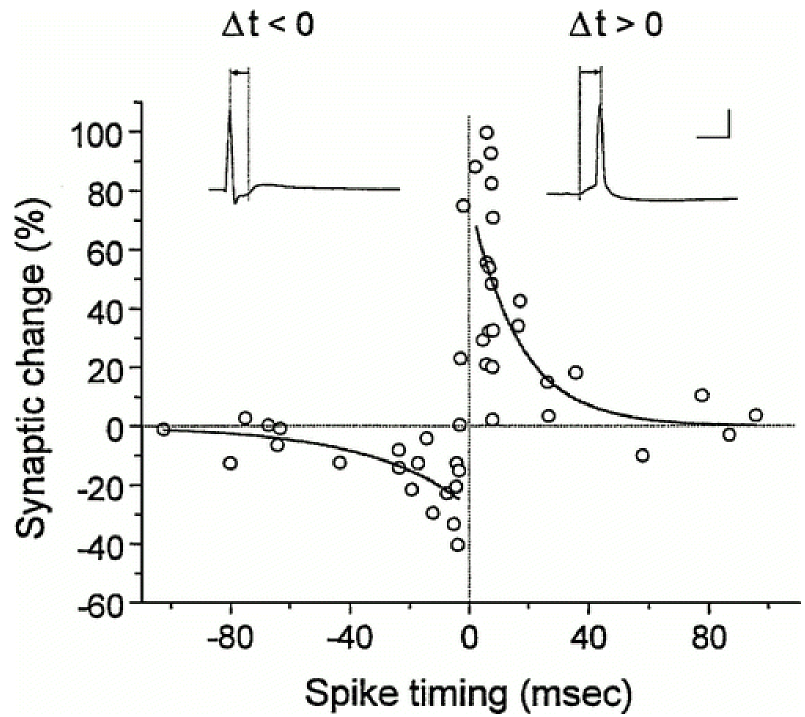
- They learn by means of synaptic change

pre-synaptic neuron

post-synaptic neuron



% synaptic modification



postsynaptic - presynaptic (ms)

Spike Time Dependent Plasticity (STDP):

- Small time window
- Strengthening (LTP) for positive time difference
- Weakening (LTD) for negative time difference

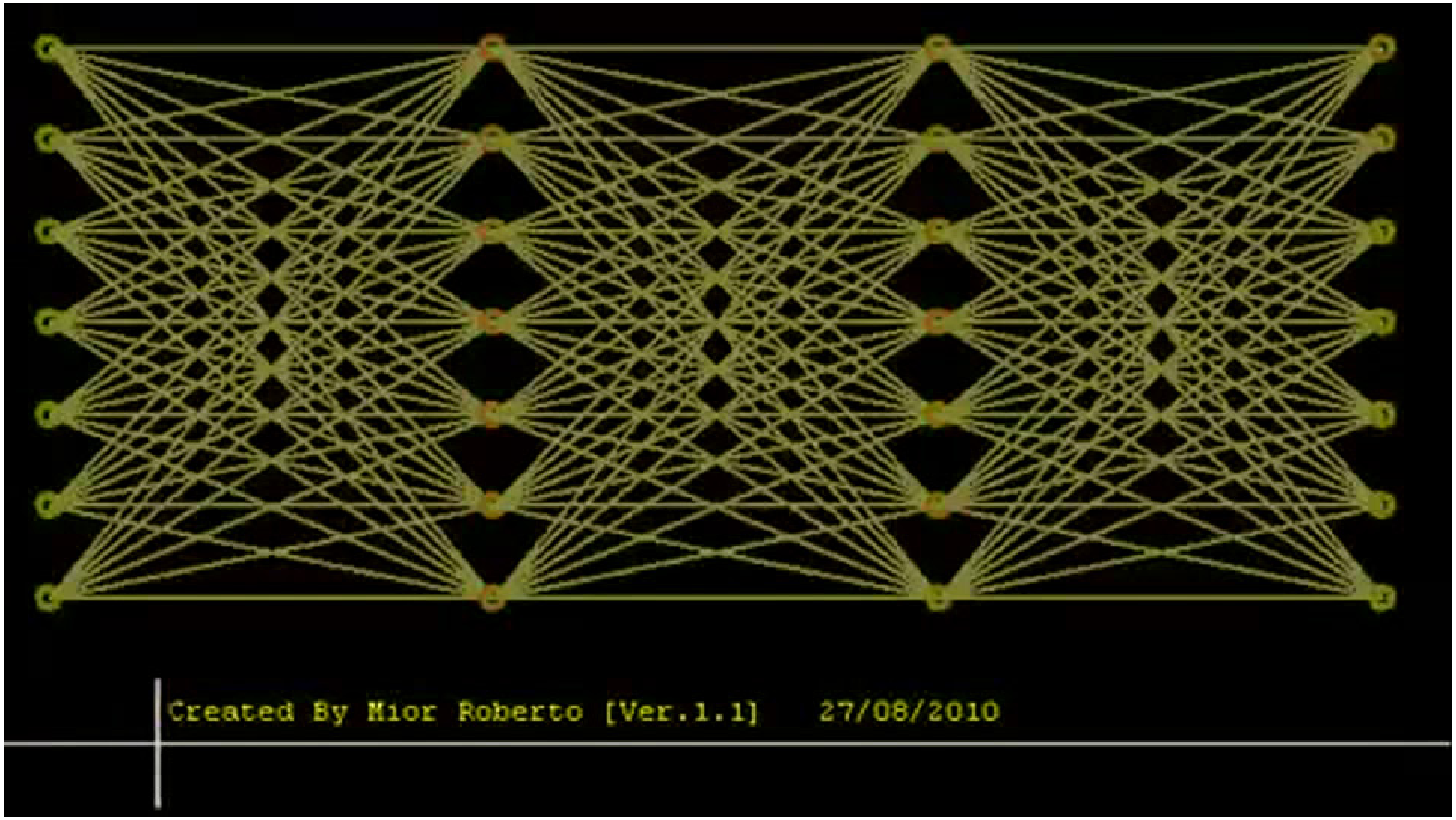
From Bi and Poo, 2001

How do Neurons Learn? (2)

- Hebb rule becomes function of time between pre- and postsynaptic neuron
- Initial synaptic weight in range e.g. *[-0.1,0.1]*
- Input pattern repeatedly presented to the network (training)
- Modification of weights after each presentation step (learning)
- Computation of synaptic weight:

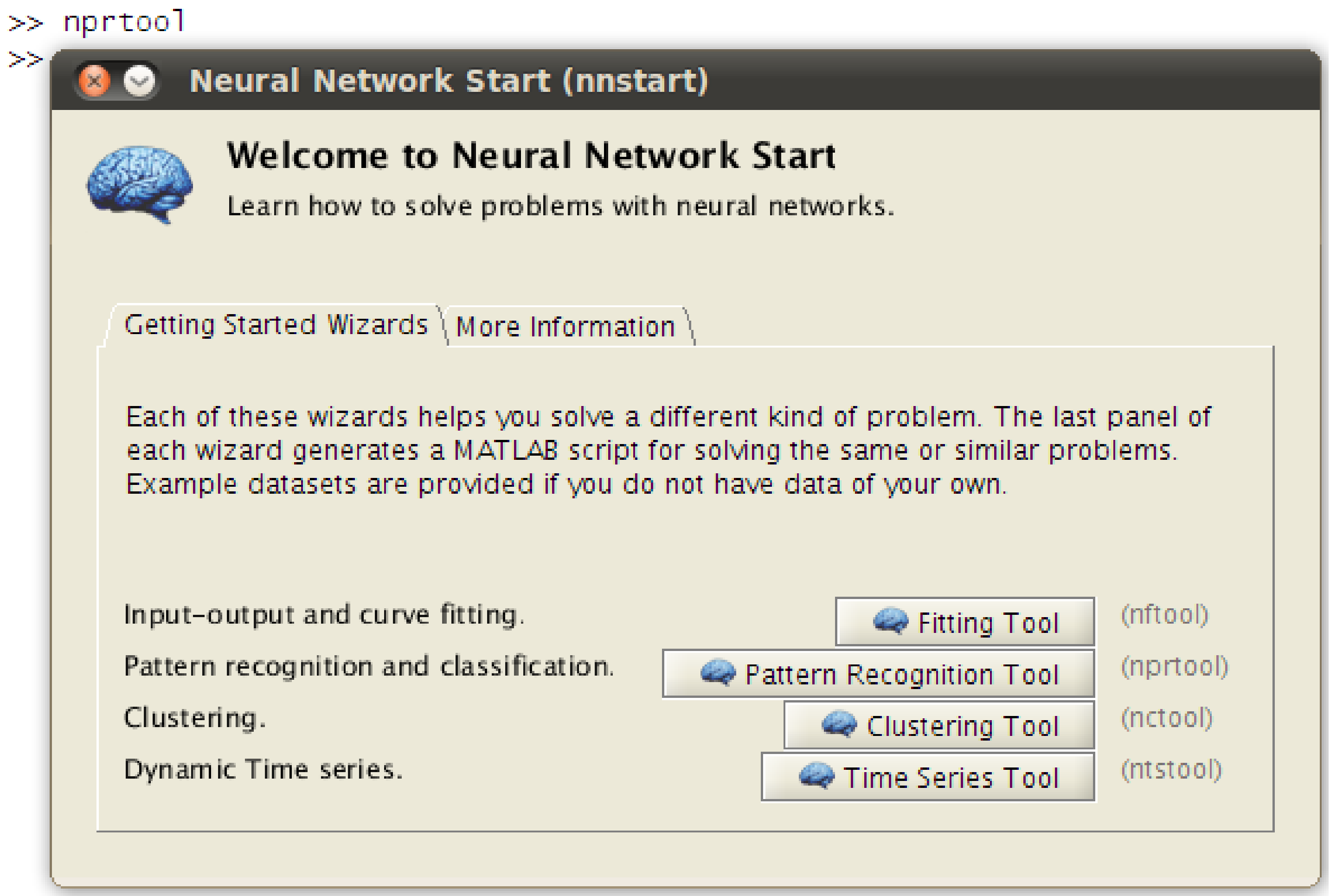
$$\omega_{ij}^{t+1} = \omega_{ij}^{t} + \Delta \omega_{ij}^{t}$$

Spiking Networks: A Simulation

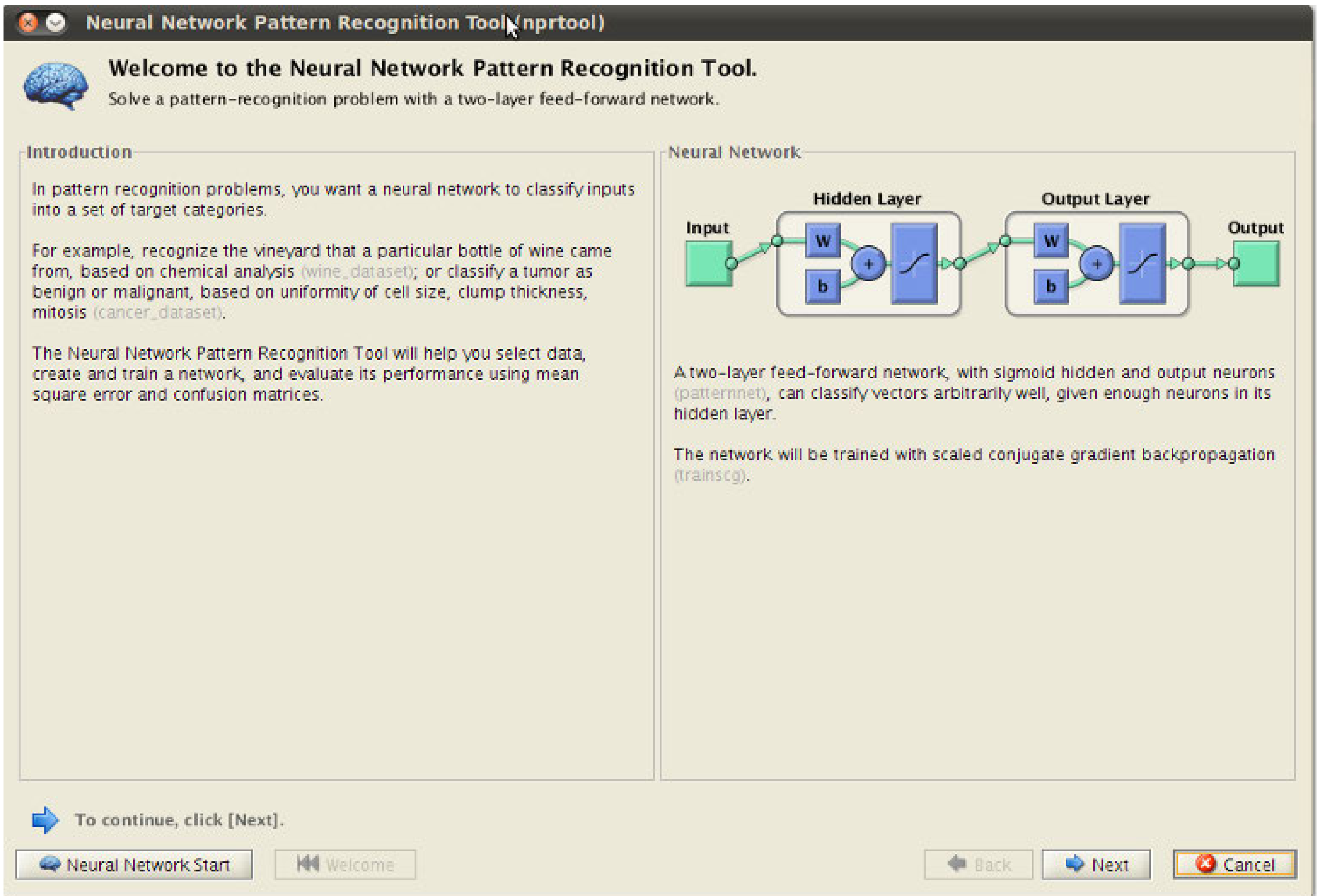


Simulation and Application: Matlab (1)

- Matlab: Neural Network Toolbox (available on all RZ-Pool-PCs)

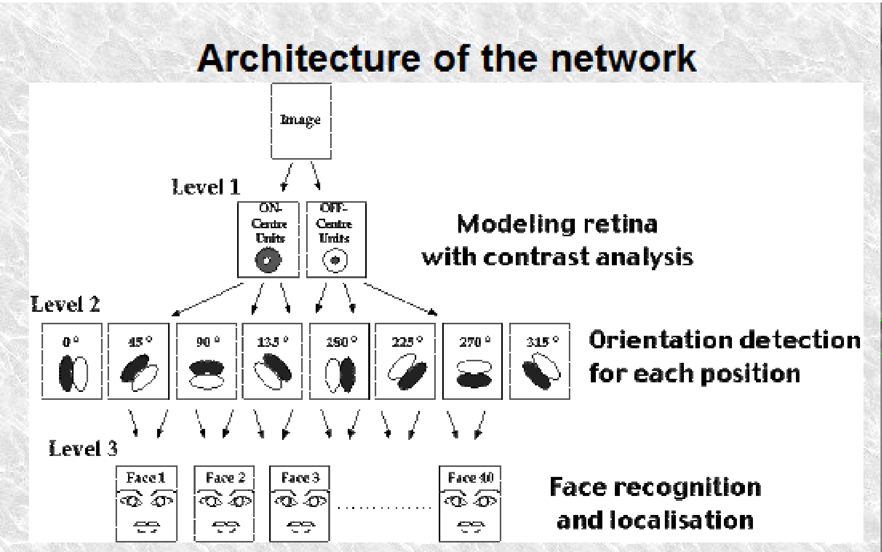


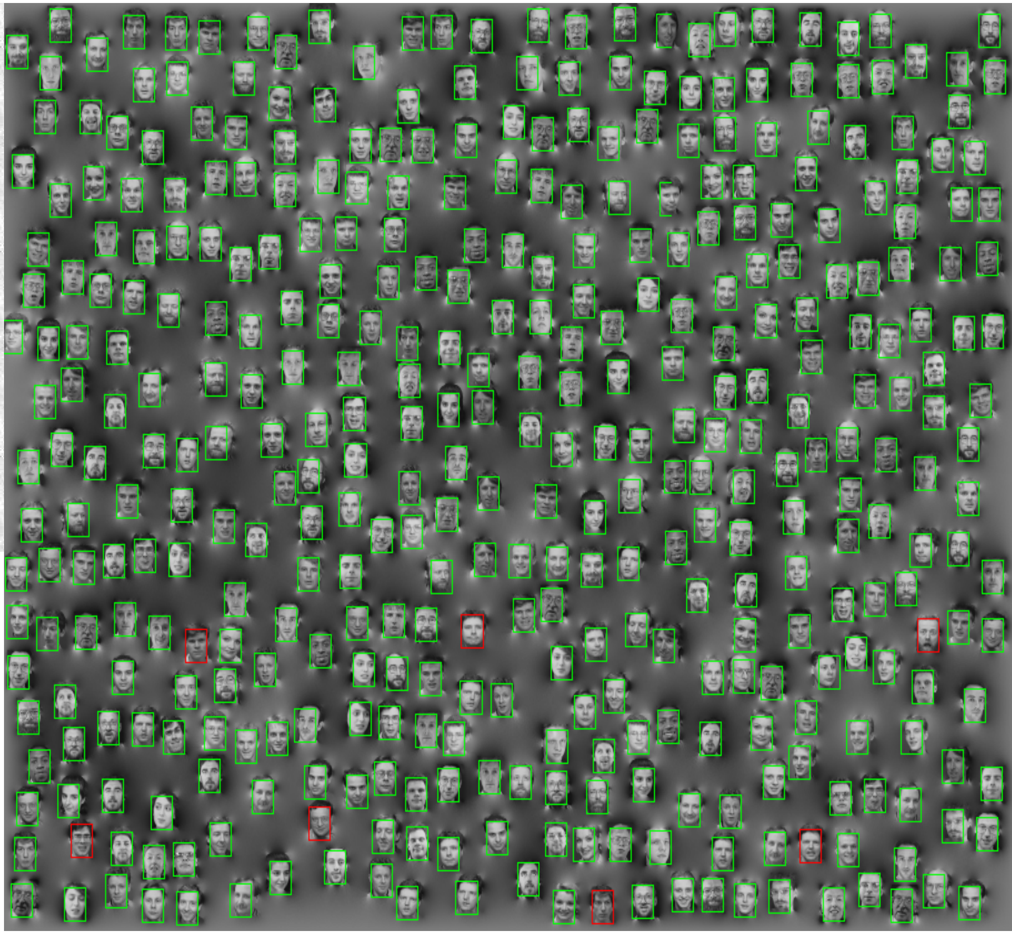
Simulation and Application: Matlab (2)



Simulation and Application: SpikeNET

- Simulator for large sets of asynchronous spiking neurons





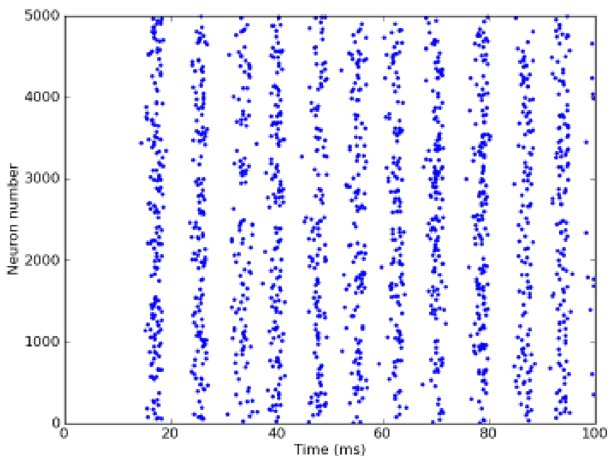
http://sccn.ucsd.edu/~arno/spikenet/

Simulation and Application: Brian

Network of sparsely connected inhibitory integrate-and-fire neurons

Dynamics of a network of sparsely connected inhibitory integrate-and-fire neurons. Individual neurons fire irregularly at low rate but the network is in an oscillatory global activity regime where neurons are weakly synchronized.
Reference: Brunel N, Hakim V, *Fast Global Oscillations in Networks of Integrate-and-Fire Neurons with Low Firing Rates*, Neural Computation 11, 1621–1671 (1999)

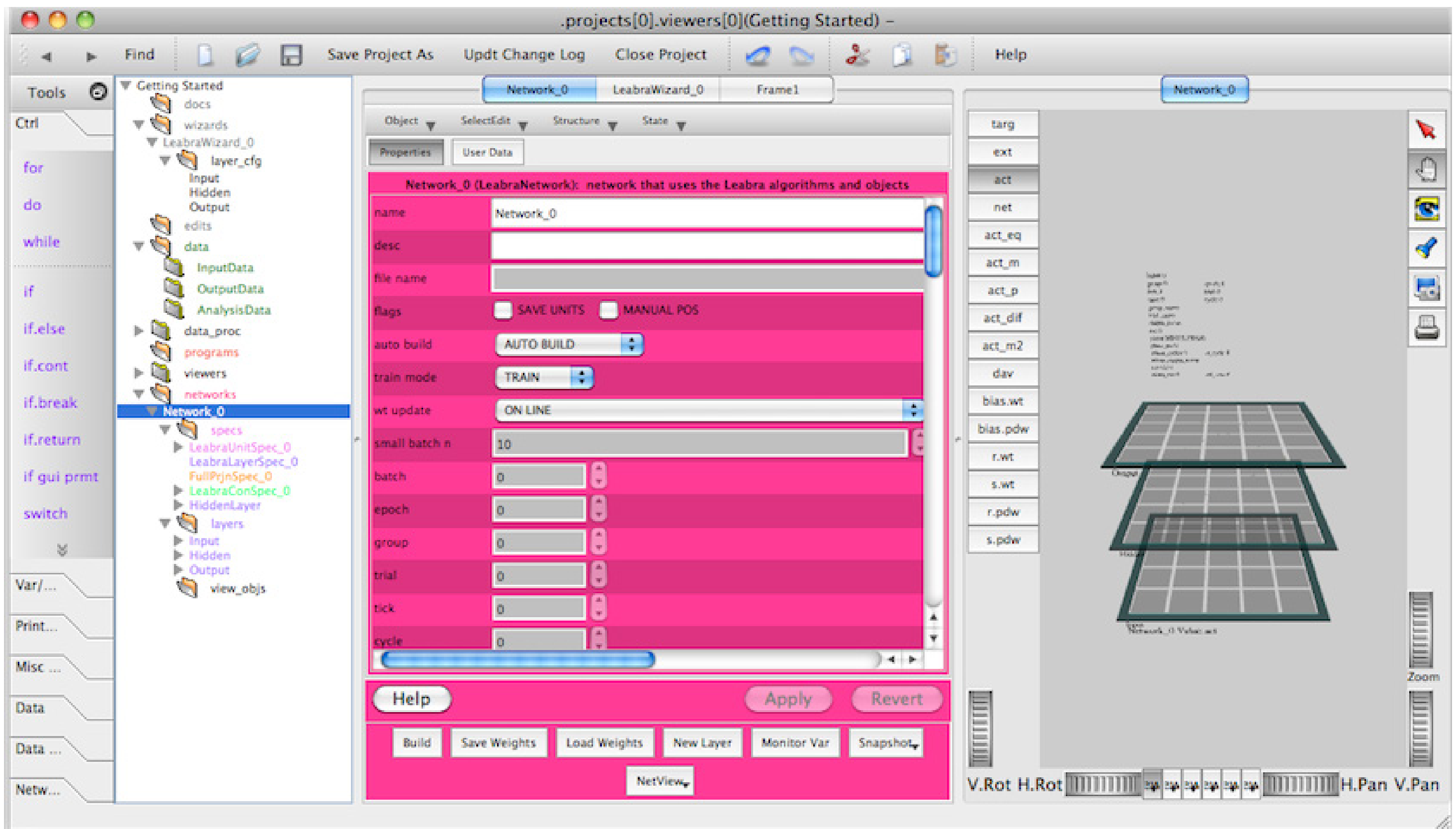
```
from brian import *
# Network parameters
N = 5000
Vr = 10 * mV
theta = 20 * mV
tau = 20 * ms
delta = 2 * ms
taurefr = 2 * ms
duration = .1 * second
C = 1000
sparseness = float(C)/N
J = .1 * mV
muext = 25 * mV
sigmaext = 1 * mV
# Neuron model
eqs = "dV/dt=(-V+muext+sigmaext*sqrt(tau)*xi)/tau : volt"
group = NeuronGroup(N, eqs, threshold=theta,
reset=Vr, refractory=taurefr)
group.V = Vr
# Connections
conn = Connection(group, group, state='V', delay=delta,
weight=-J, sparseness=sparseness)
# Monitors
M = SpikeMonitor(group)
# Run
run(duration)
# Plot
raster_plot(M)
show()
```



http://www.briansimulator.org/

Simulation and Application: emergent

- Supports Backpropagation (feedforward and recurrent), Self-Organizing (e.g., Hebbian, Kohonen, Competitive Learning), Constraint Satisfaction (e.g., Boltzmann, Hopfield) in one coherent, biologically-plausible framework



http://grey.colorado.edu/emergent/index.php/Getting_Started

Simulation and Application: GENESIS

```
#!/usr/local/bin/genesis-g3
#!

create cell /n
create segment /n/soma

model_parameter_add /n/soma Vm_init -0.068
model_parameter_add /n/soma CM 0.0164
model_parameter_add /n/soma RM 1.500
model_parameter_add /n/soma RA 2.500
model_parameter_add /n/soma ELEAK -0.080

model_parameter_add /n/soma LENGTH 4.47e-5
model_parameter_add /n/soma DIA 2e-5

runtime_parameter_add /n/soma INJECT 2e-9
output_add /n/soma Vm

run /n 0.05

quit
```

- Modelling of single and multiple compartment models
- Under development since 1988, current version no.3
- Code example for simple single compartment model:
- Use interactivly within a UNIX-shell

```
$ genesis-g3
Welcome to the GENESIS 3 shell
genesis >
```

http://genesis-sim.org/

Simulation and Application: Some more links

- http://www-ist.massey.ac.nz/smarsland/MLbook.html

Accompanying source code in Python to the introductory book in Machine Learning

- http://www.neuron.yale.edu/neuron/

Simulation framework for empirically-based modelling; comes along with lists of publication on research in the field of computational neuroscience and provides model databases

- http://pybrain.org/

Python-based modular library providing algorithms for Neural Networks and Training strategies

Summary

- Spiking Neural Networks resemble biological information transmission
- Temporal coding opposed to static learning (input determines output, distance measures in feature space)
- Learning based on synaptic correlations (Hebb Rule)
- Concept of Spike-Time Dependent Plasticity
- Different models: Hodgkin and Huxley, Integrate and Fire, Spike-Response
- Applications: Pattern recognition (speech, face detection,..)
- Software implementations: SpikeNET, Brian, GENESIS,…

Further Readings

- Floreano, Mattiussi book, chapter 3, most material covered in this lecture; detailed info in 2 books

- Optional details
  - Gerstner and Kistler, 2002, Spiking Neuron Models
  - Maass, Bishop 2001, Pulsed Neural Networks