1. From Neurons to Networks

From Neurons to Networks

biological neurons (very brief)
- single neurons
- synapses and networks
- synaptic plasticity and learning

simplified description
- modelling of biological neurons/networks
- **inspiration for artificial neural networks**

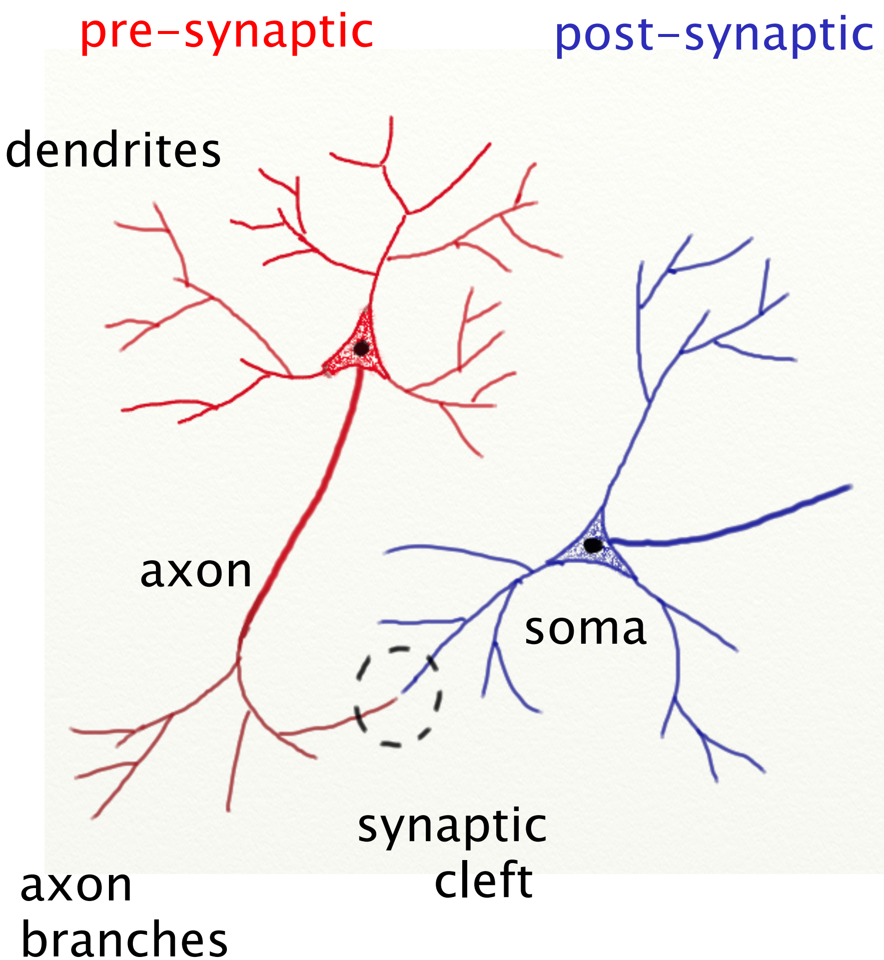
artificial neural networks
- unrealistic simplifications
- architectures and types of networks:

  recurrent **attractor neural networks** (associative memory)

  Example: **The Hopfield model**

  feed-forward neural networks (classification/ regression)

From Neurons to Networks



neurons:

highly specialized cells

- cell body **soma**
- incoming **dendrites**
- branched **axon**

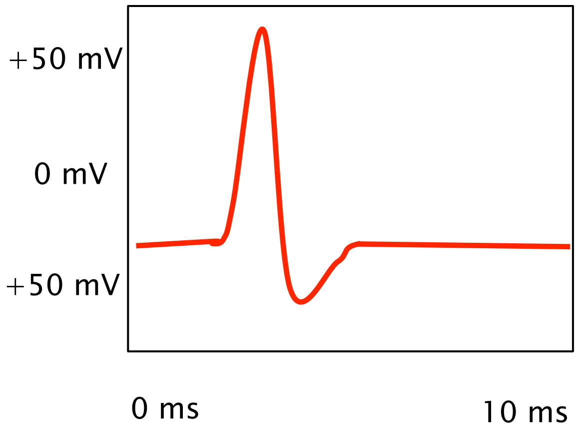
many neurons !

$\gtrsim 10^{12}$ in the brain

highly inter-connected !

$\gtrsim 1000$ *neighbors*

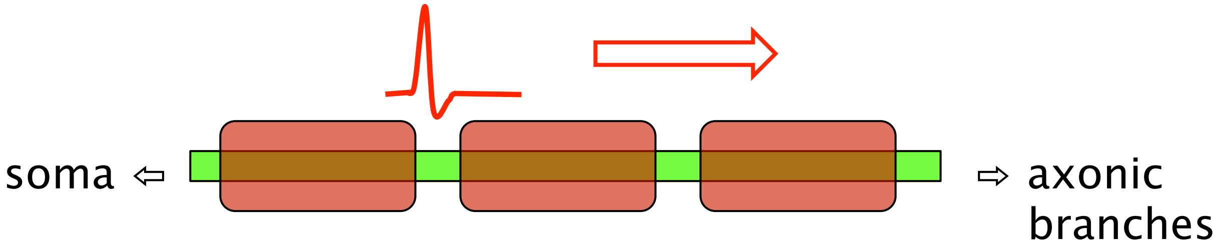
action potentials / spikes:



- cells generate localized electric pulses
  membrane is charged / discharged
  by means of ion transport

- class of models (electro-chemistry):
  Hodgkin-Huxley neurons + extensions

- signals travel along the axon
  (serves as *cable*)



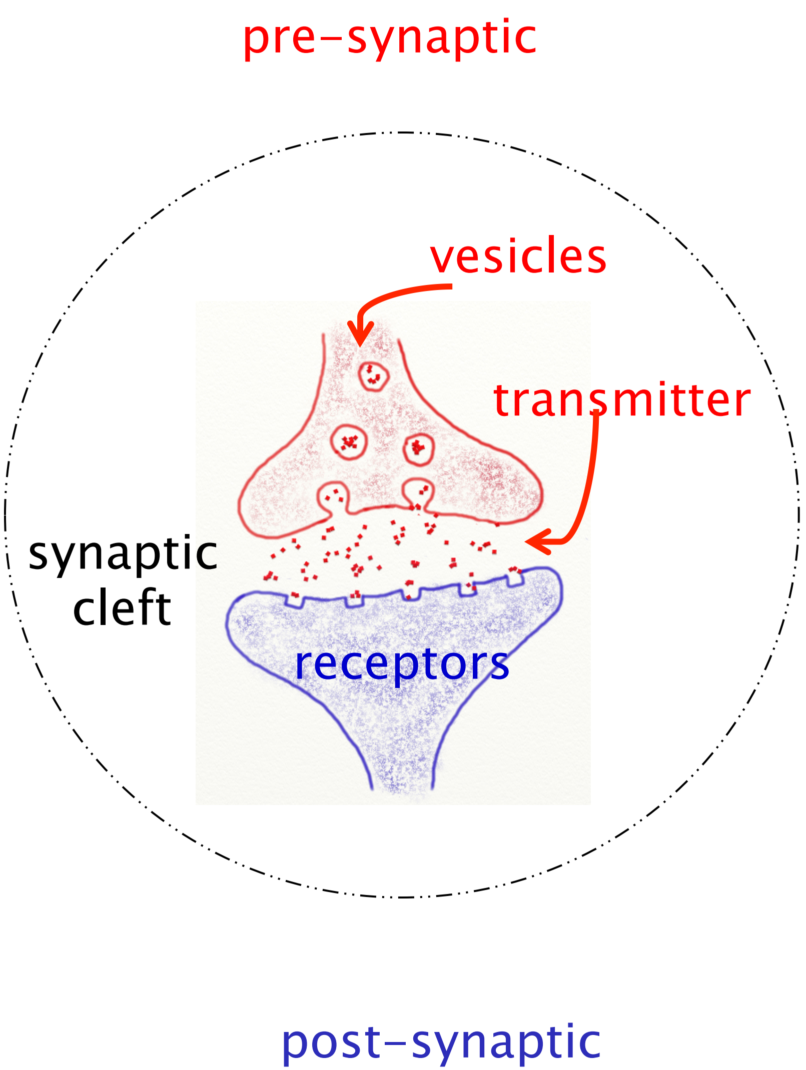
From Neurons to Networks

synapse:

- arriving pre-synaptic spike triggers release of **neuro-transmitters**

- different kinds of synapses play specific roles:

- pre-synaptic pulse at **excitatory /inhibitory** synapse triggers / hinders post-synaptic spike generation



university of groningen

From Neurons to Networks

- incoming pulses $\begin{cases} \text{increase} & & \text{(excitatory)} \\ & \text{membrane potential} & \\ \text{decrease} & & \text{(inhibitory)} \end{cases}$

- all or nothing response

potential exceeds **threshold** ⇨ postsynaptic neuron *fires*

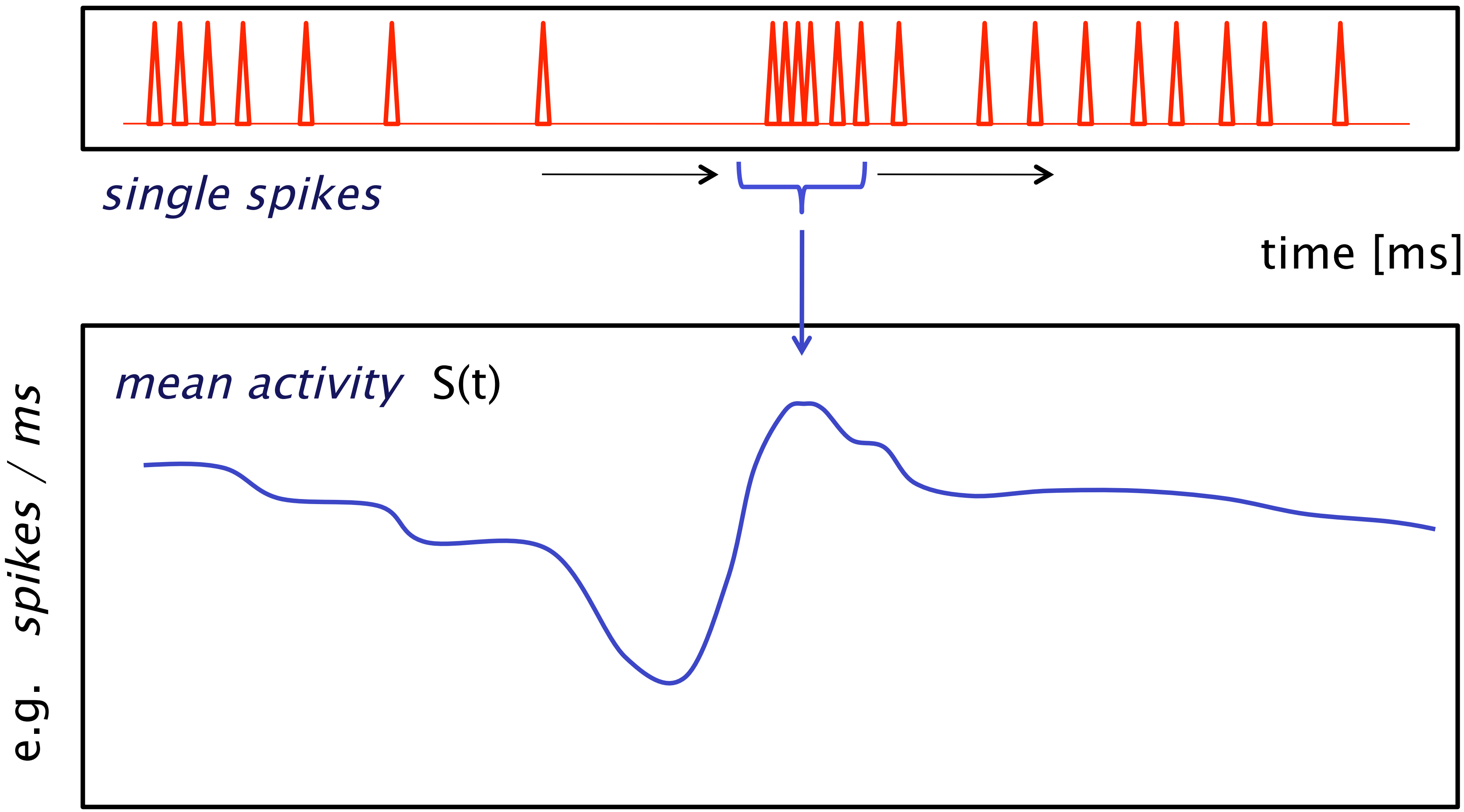
potential is sub-**threshold** ⇨ **postsynaptic neuron** *rests*

- important class of models:
  Integrate-and-Fire neurons and networks
  - no details of spike generation and signal propagation
  - neuron as simple threshold element,
  - excitatory and inhibitory connections with other units

(see additional material)

From Neurons to Networks

simplified description of neural activity: **firing rates**



From Neurons to Networks

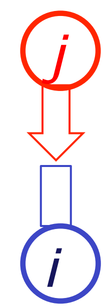
(mean) **local potential** at neuron *i* (with activity S_i)

$$\sum_j w_{ij} S_j$$

weighted sum of incoming activities

synaptic weights

$$w_{ij} = \begin{cases} > 0 & \textbf{excitatory}\text{ synapse} \\ = 0 & \text{no incoming synapse} \\ < 0 & \textbf{inhibitory}\text{ synapse} \end{cases}$$



Note: $w_{ii} = 0, \quad w_{ij} \neq w_{ji}$ (in general)

connections *i→j and j→i* are unrelated.

Activation Function

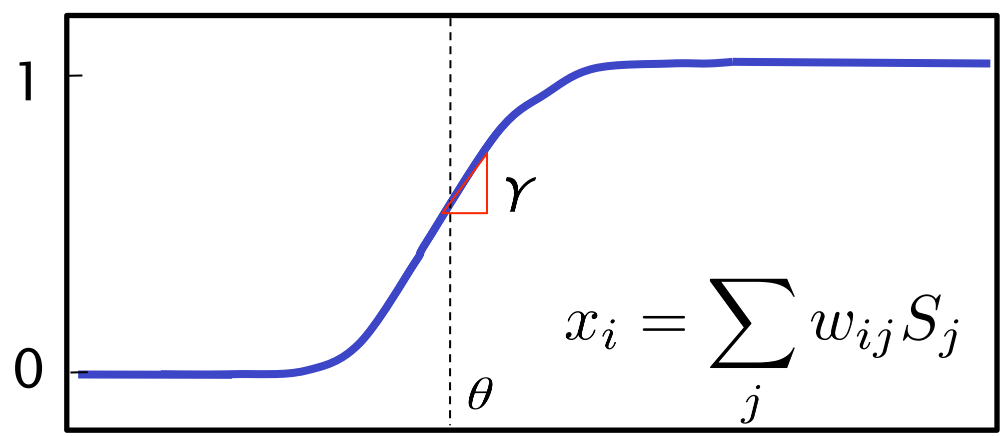
non-linear response: $S_i \; = \; h\left[\sum_j w_{ij} S_j\right]$

- minimal activity $\quad h(x \rightarrow -\infty) \equiv 0$
- maximal activity $\quad h(x \rightarrow +\infty) \equiv 1$
- monotonic increase $\quad h'(x) > 0$

important class of fcts.:
sigmoidal activation

just one example: $$h(x_i) = \frac{1}{2}\Big(1 + \tanh\left[\gamma(x_i - \theta)\right]\Big)$$

gain parameter γ
local threshold θ
Remark:
do not confuse θ *with threshold for single spike generation!*



Symmetrization (I)

non-linear response: $S_i = g\left[\sum_j w_{ij} S_j\right]$

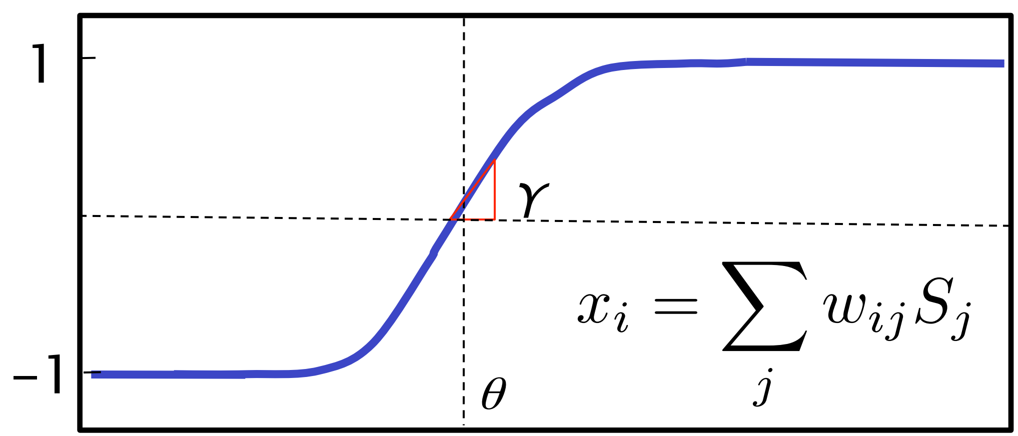
- minimal activity $g(x \rightarrow -\infty) \equiv -1$
- maximal activity $g(x \rightarrow +\infty) \equiv 1$
- monotonic increase $g'(x) > 0$

sigmoidal activation

just one example: $g(x_i) = \tanh\left[\gamma(x_i - \theta)\right]$

gain parameter γ

local threshold θ



Symmetrization (I)

$$-1 \leq S_i = g\left[\sum_j w_{ij} S_j\right] \leq +1$$

$S_j > 0, w_{ij} > 0$	**excitation**	$S_j < 0, w_{ij} < 0$
$S_j > 0, w_{ij} < 0$	**inhibition**	$S_j < 0, w_{ij} > 0$
(plausible)		**(unrealistic)**

symmetric treatment of low / high activity

- not suitable for realistic modelling of biological neurons, absence of spikes cannot excite activity etc. (further consequences discussed below)
- convenient in bio-inspired **artificial neural networks** (more drastic simplifications to come)

McCulloch Pitts Neurons

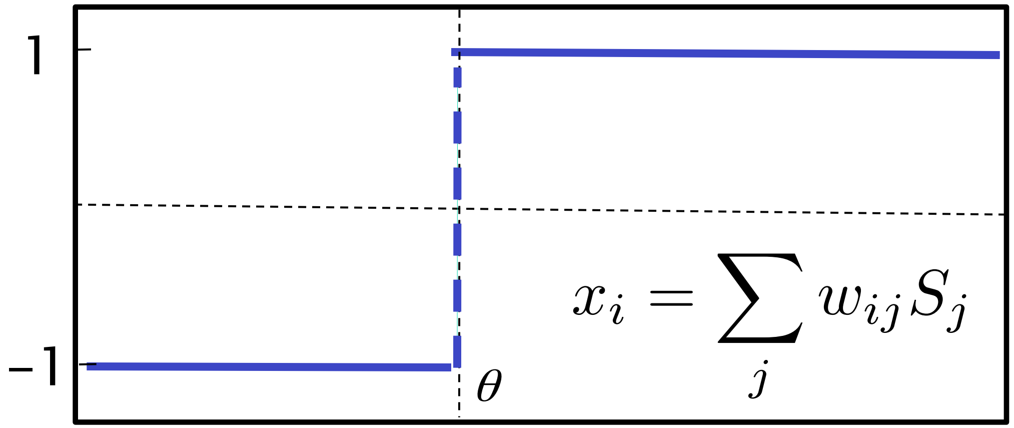
an extreme case: **infinite gain** $\gamma \to \infty$

$$g(x_i) = \tanh\left[\gamma(x_i - \theta)\right] \to \mathrm{sign}\left[x - \theta\right] = \begin{cases} +1 & \text{for } x \geq \theta \\ -1 & \text{for } x < \theta \end{cases}$$

McCulloch Pitts [1943]:
model neuron is either quiescent or maximally active
do not consider graded response

local threshold θ

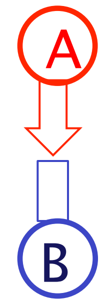
Remark:
this is not the
all–or–nothing response
in single spike
generation!



Synaptic Plasticity

D. Hebb [1949]

consider – presynaptic neuron A
– postsynaptic neuron B
– excitatory synapse w_{BA}



Hypothesis: **Hebbian Learning**

If A and B (frequently) fire at the same time the excitatory synaptic strength w_{BA} increases

→ memory-effect will favor joint activity in the future
high activity of A will trigger high activity at B
correlations of activity stored in the network

Symmetrization (II)

$$-1 \leq S_A, S_B \leq +1$$

change of synaptic strength $\Delta w_{BA} \propto S_A S_B$

beyond Hebb's hypothesis: **symmetric treatment**

- of high/low activity
- of pre- postsynaptic activity

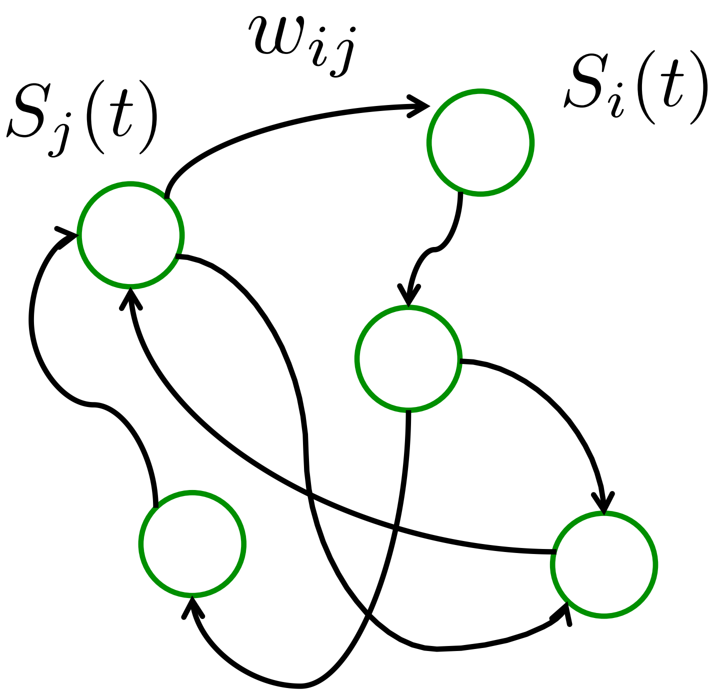
S_A	S_B		
high	high	strengthens	excitatory
low	low	weakens	inhibitory
high	low	weakens	excitatory
low	high	strenghtens	inhibitory

synapse w_{BA}

Artificial Neural Networks

in the following:

- assembled from simple *firing rate neurons*
- connected by *weights*, real valued synaptic strenghts
- various architectures and types of networks
  e.g.: **attractor neural networks, feed-forward networks**



here:

number of neurons N=5

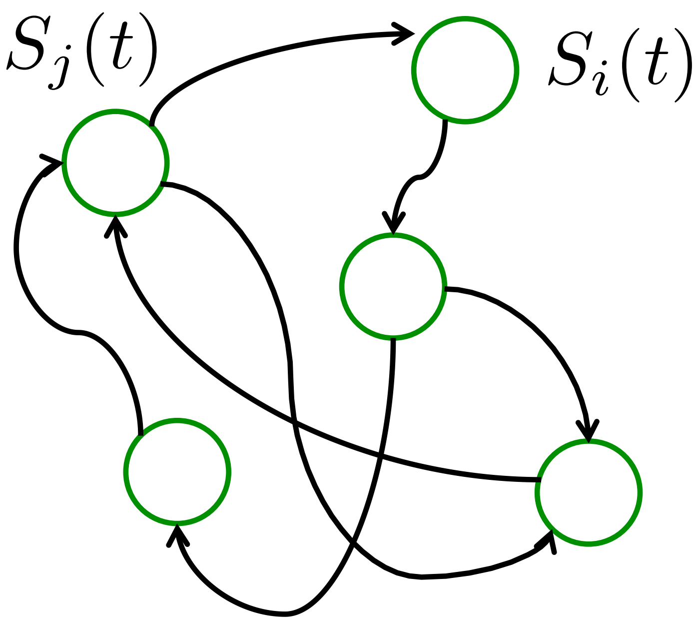
partial connectivity:

7 directed synapses

Attractor Neural Networks

training phase:
dynamics of weights (synapses), given the network activity

$$w_{ij}(t+\Delta t) = w_{ij} + \eta S_i(t) S_j(t)$$ **(Hebbian Learning)**

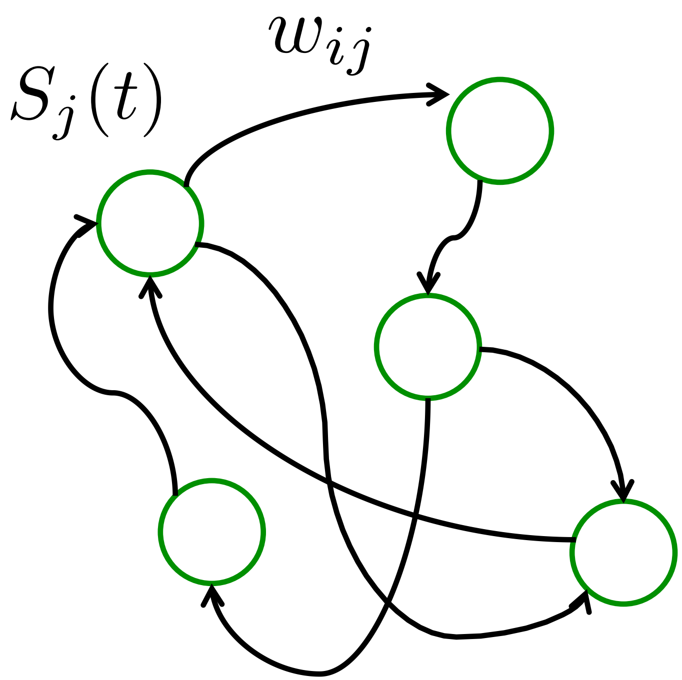


for example:
sequence of **activity patterns,**
memorization of frequent states,
external stimuli etc.

$$\mathbf{S}(t) \in \{\boldsymbol{\xi}^\mu\}_{\mu=1}^P$$

Attractor Neural Networks

working phase:
dynamics of neurons, given the weights



$$S_i(t+\Delta t) = g\left[\sum_j w_{ij} S_j(t)\right]$$

variants:

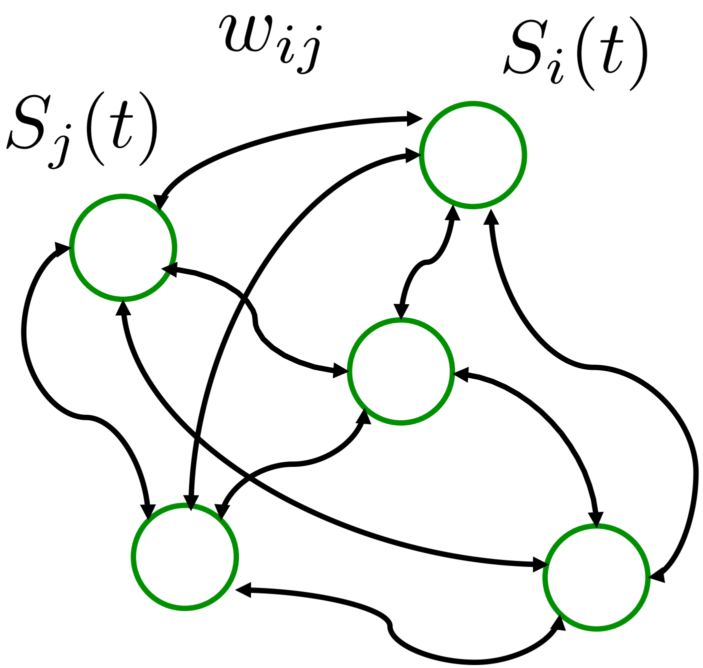
- sequential update $\quad i=1,2,3,\ldots$
- random sequential update
- parallel update

retrieval of patterns: from incomplete / noisy state $\mathbf{S}(0)$

approach original pattern $\mathbf{S}(t\to\infty) = \boldsymbol{\xi}^{\nu}$

extreme example: **Little-Hopfield model**

- symmetric Hebbian interactions $$w_{ij} = w_{ji} = \frac{1}{P} \sum_{\mu=1}^{P} \xi_i^{\mu} \, \xi_j^{\mu}$$



- McCulloch Pitts neurons

$$S_i(t+\Delta t) = \mathrm{sign}\left[\sum_j w_{ij} S_j(t)\right]$$

- parallel (Little)
  random sequential (Hopfield)
  dynamics of neurons

consider a single *pattern* $\boldsymbol{\xi}^1$ $\quad w_{ij} \propto \xi_i^1 \xi_j^1$

show: $\quad \mathbf{S}(t) = \boldsymbol{\xi}^1$ is reproduced under the dynamics!

$$S_i(t+\Delta t) = \mathrm{sign}\left(\sum_j w_{ij}\xi_j^1\right) = \mathrm{sign}\left(\xi_i^1 \sum_j \underbrace{\xi_j^1 \xi_j^1}_{>0}\right) = \xi_i^1$$

fixed point: unchanged under system dynamics

(?) stable fixed point: small deviations from patterns vanish

(?) attractor: retrieved from similar initial states

much more difficult to show: set of random (zero mean) patterns

$\{\boldsymbol{\xi}^\mu\}_{\mu=1}^P$ with $P = \alpha N$ is retrieved, if α is not too big

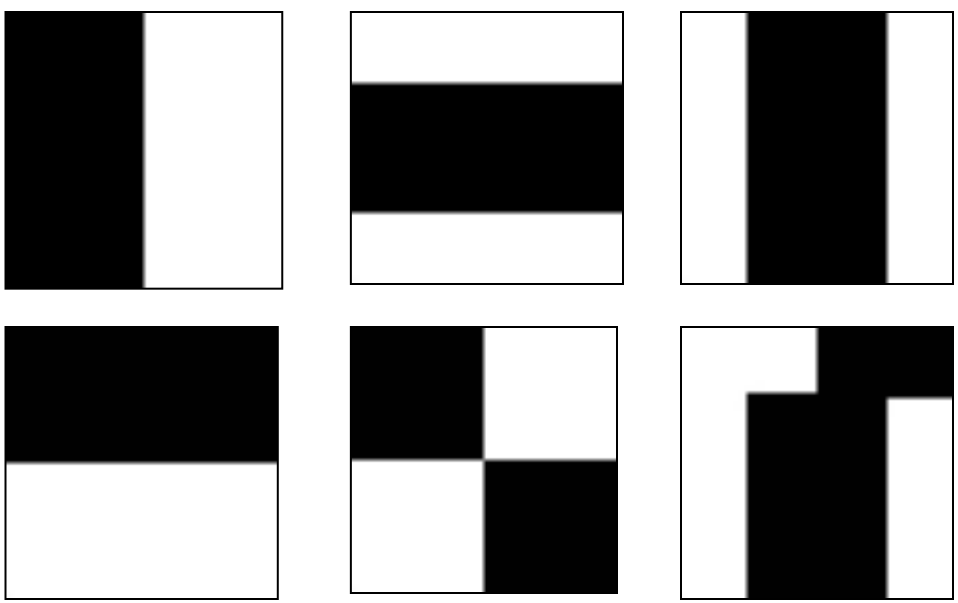
N neurons (with ~*N*² weights) *memorize* ~$\mathcal{O}(N)$ activity patterns university of groningen

Demo: Associative Memory

- consider a **Hopfield network** of 2500 neurons
- visualized as array of 50*50 neurons

  note: there is no spatial structure imposed, every neuron is connected with every other neuron!

training patterns:



+ (994) random patterns with activity +/- 1 with equal probability

saved as 'pattern.mat'

Demo: Associative Memory

- calculate synaptic matrices
  saved in jmats.mat
  (load by `load jmats.mat`)

$$w_{ij} = w_{ji} = \frac{1}{P} \sum_{\mu=1}^{P} \xi_i^\mu \, \xi_j^\mu$$

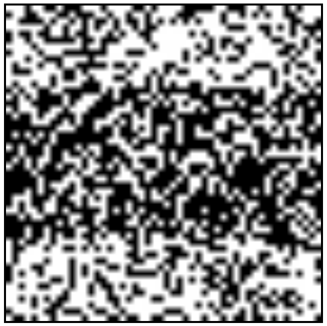
P=10 (jmat10)
P=100 (jmat100)
P=1000 (jmat1000)

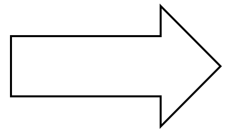
hopf (jmat, inipat, q, stps) e.g. hopf(jmat10,1,0.4,100)

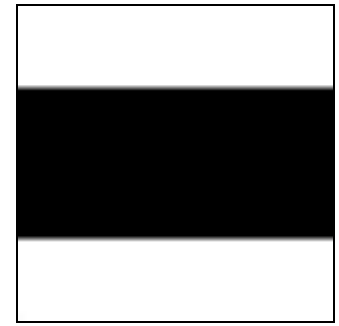
– jmat	synaptic matrix to be used
– inipat	index of pattern used as initial state
– q	fraction of flipped spins in initial state
– stps	number of steps per neuron in the system

Demo: Associative Memory

regime of low 'memory load' (here: *P=10*)







noisy initial state

dynamics

(almost) perfect retrieval

- distributed memory
- robust system!

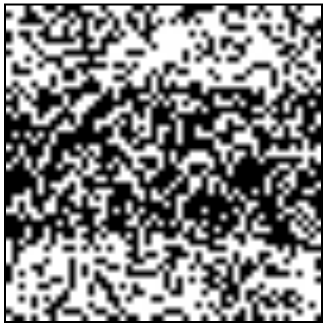
one can show: set of random (zero mean) patterns

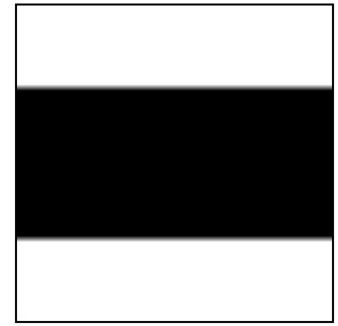
$\{\boldsymbol{\xi}^{\mu}\}_{\mu=1}^{P}$ with $P = \alpha N$ is retrieved *successfully* for $\alpha \leq 0.14$

N neurons (with $\sim N^2$ weights) *memorize* $\sim \mathcal{O}(N)$ activity patterns

Demo: Associative Memory

intermediate load (e.g. *P=100*)





noisy initial state

- distributed memory
- robust system!

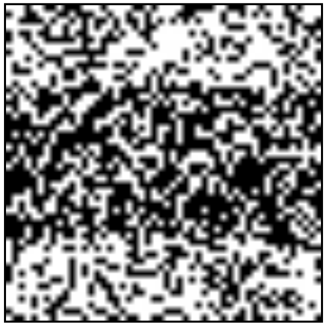
dynamics

noise dependent
retrieval state
potential 'confusion'

one can show: besides the original patterns,
'spurious states' are also stabilized,
e.g. mixtures of different patterns

Demo: Associative Memory

overloaded memory (e.g. $P=1000$)



?

noisy initial state

dynamics

no retrieval
patterns unstable

- distributed memory
- robust system!

Remarks

- the above is, strictly speaking, only true for 'uncorrelated patterns'

- one can improve the Hopfield network by *decorrelating,* i.e.

$$w_{ij} = \frac{1}{P} \sum_{\mu=1}^{P} \Big(\xi_i^\mu - \langle \xi_i^\mu \rangle \Big) \Big(\xi_i^\mu - \langle \xi_j^\mu \rangle \Big)$$

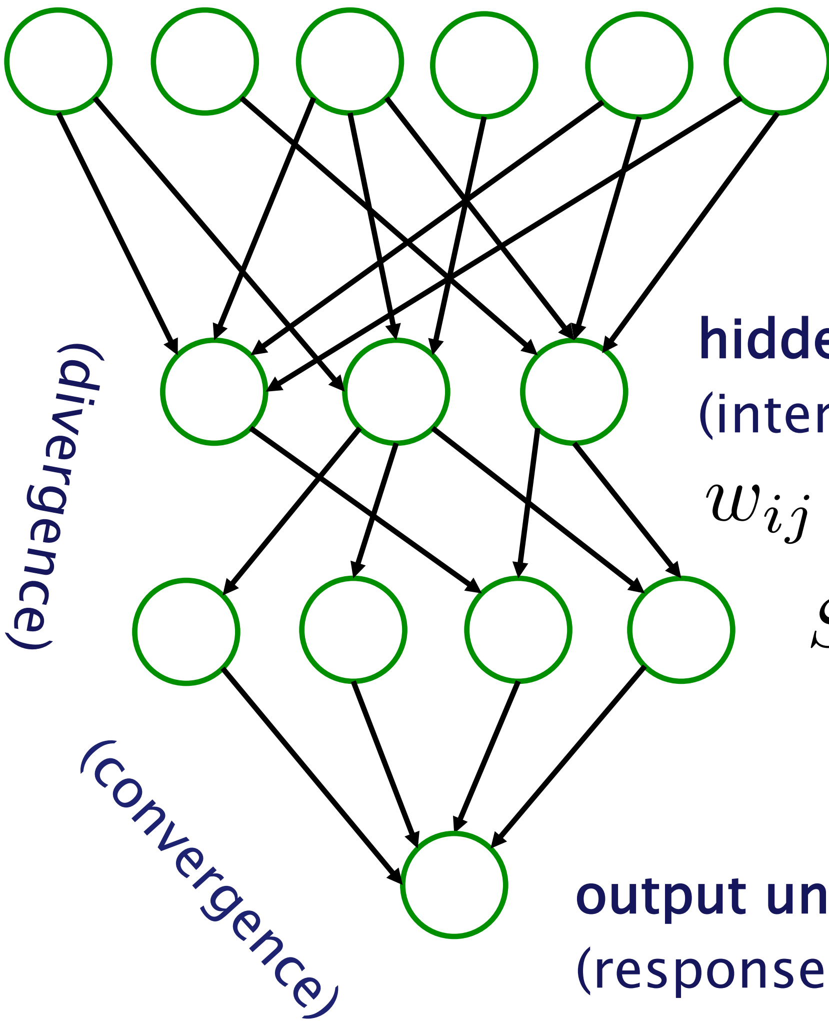
- improved training prescriptions:
  iterative learning, e.g. perceptron algorithm

- Hopfield model serves as a basic model of learning and retrieval
- other 'recurrent networks' are used in applications
- more sophisticated and detailed models for biological modelling

- different architectures, e.g.
  'Feed-forward' networks for regression/classification tasks

Feed–Forward Networks

input layer (external stimulus)



hidden units
(internal representation)

w_{ij}

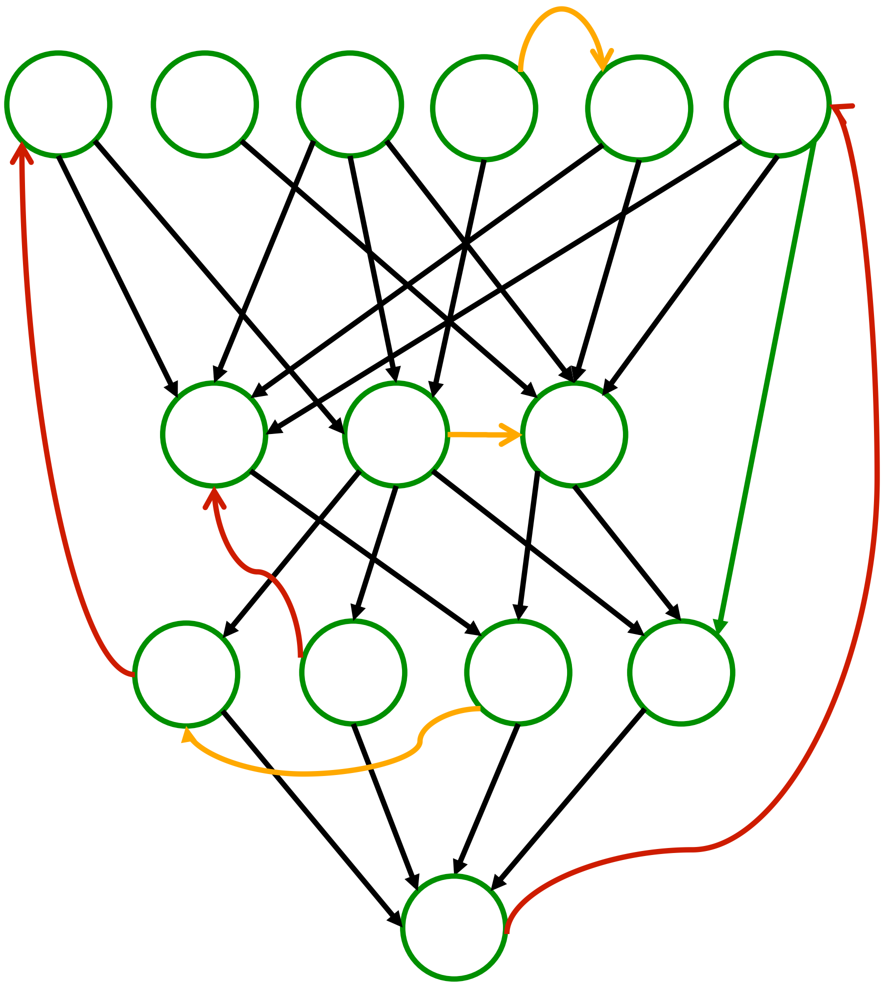
$$S_i = g\left(\sum_j w_{ij} S_j\right)$$

output unit(s)
(response to the stimulus)

layered architecture
(here: 6–3–4–1)

directed connections
(here: only to next layer)

Feed-Forward Networks



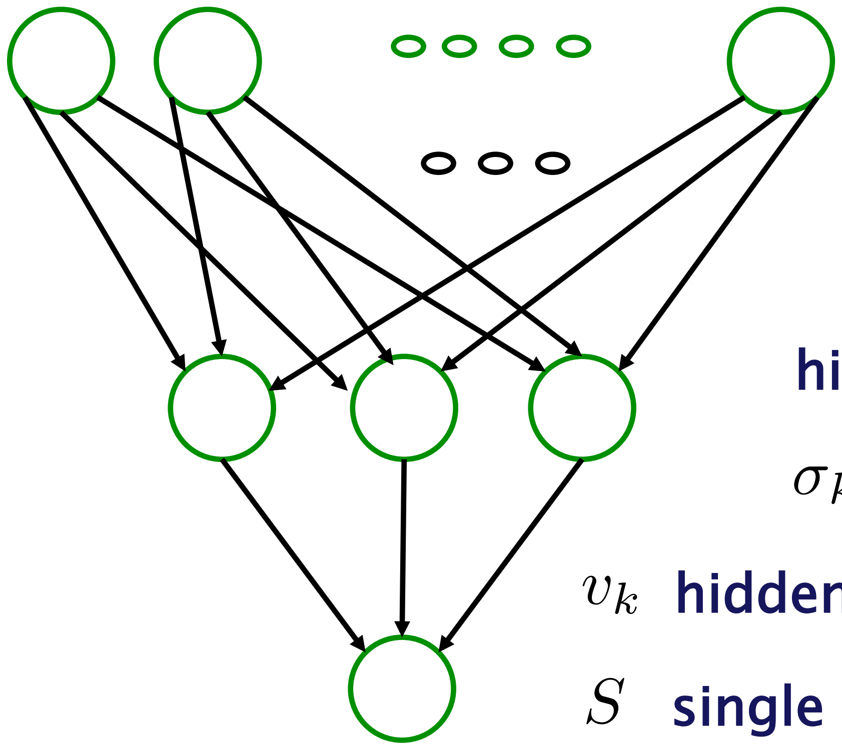
modifications:

(still) feed-forward weights
input/output relation

intra-layer connections
internal dynamics

feed-back connections
recurrent networks,
intermediate connectivity

Feed-Forward Networks



input units $\xi_j \in I\!R$

$w_j^{(k)}$ input to hidden weights

hidden layer units

$\sigma_k = g\left(\sum_j w_j^{(k)} \xi_j\right)$

v_k hidden to output weights

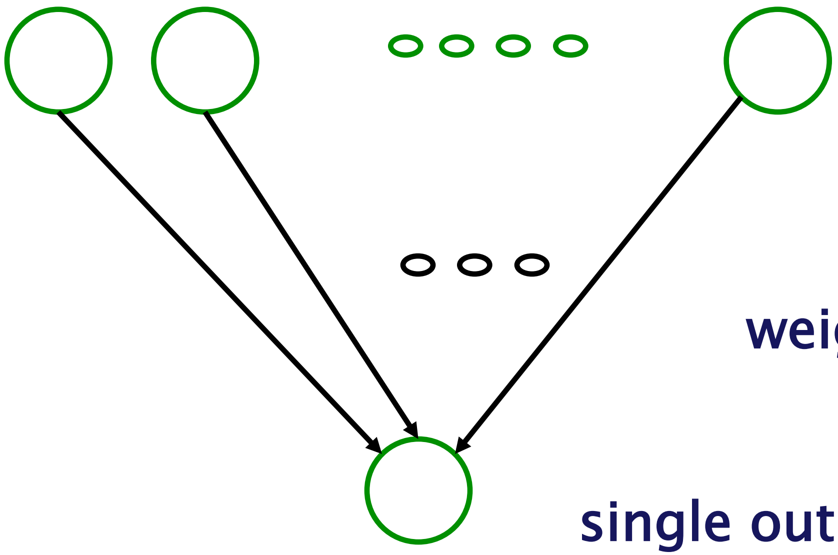
S single output unit

output = **non-linear function** of input variables:

$$S = g_{out}\left(\sum_{k=1}^{K} v_k \sigma_k\right) = g_{out}\left(\sum_{k=1}^{K} v_k \, g\big(\sum_j w_j^{(k)} \xi_j\big)\right)$$

parameterized by set of all weights (and threshold)

The Perceptron



input units $\xi_j \in I\!R$

weights $w_j \in I\!R, \mathbf{w} \in I\!R^N$

single output unit

$$S = \text{sign}\left(\sum_{j=1}^{N} w_j \xi_j - \theta\right)$$

output = *"linear separable function"* of input variables

parameterized by the weight vector $\mathbf{w}$ and threshold θ

Feed-Forward Networks

training phase:
adaptation of parameters
based on a set of example data $D = \{\boldsymbol{\xi}^\mu, S^\mu\}_{\mu=1}^P$

working phase
generalization, i.e. appl. of hypothesis to novel input $\boldsymbol{\xi} \rightarrow S$

possible aims:

classification: S is a discrete label (e.g. $S \in \{1, 2, \ldots, C\}$)
which assigns vectors ξ to one of **C** classes

regression: S is a continuous response (e.g $S \in I\!R$)
which describes a property of ξ quantitatively

Feed-Forward Networks

supervised learning (e.g. in feedforward networks)

classification / **regression** problems

- which are difficult to formulate as a simple s*et of rules* (unknown or too complex)

- for which it is relatively simple to obtain *example data* (by observation or consulting an expert)

just a few examples:

- handwritten character (digit) recognition
- prediction of protein structure from amino acid sequences
- analysis of spectra (e.g. infrared spectra -> fat content of meat)
- prediction of customer interests (e.g. at amazon.com)
- adaptive engine control (driver dependent)
- medical data/image analysis (diagnosis, risk prediction etc.)
- fault identification in technical processes

. . . many more . . .