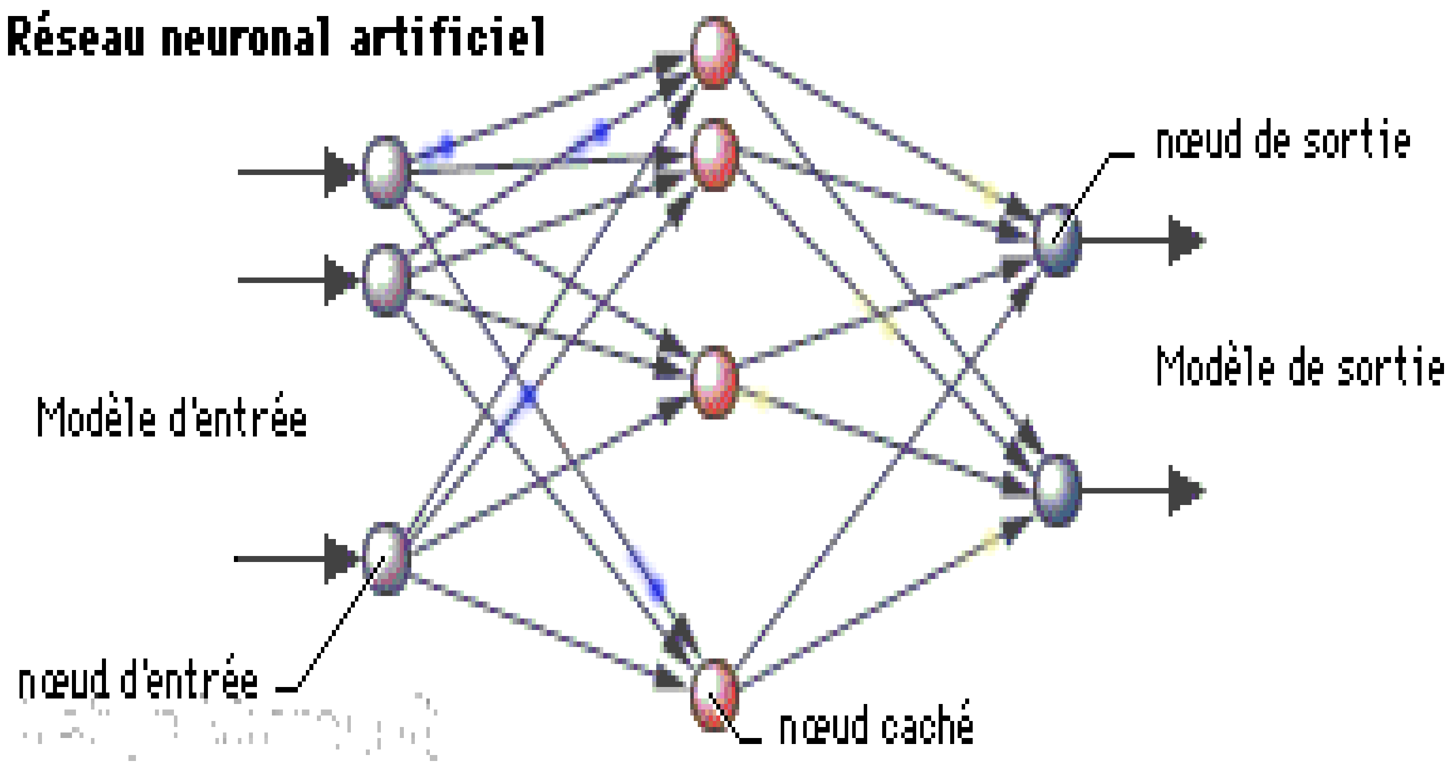
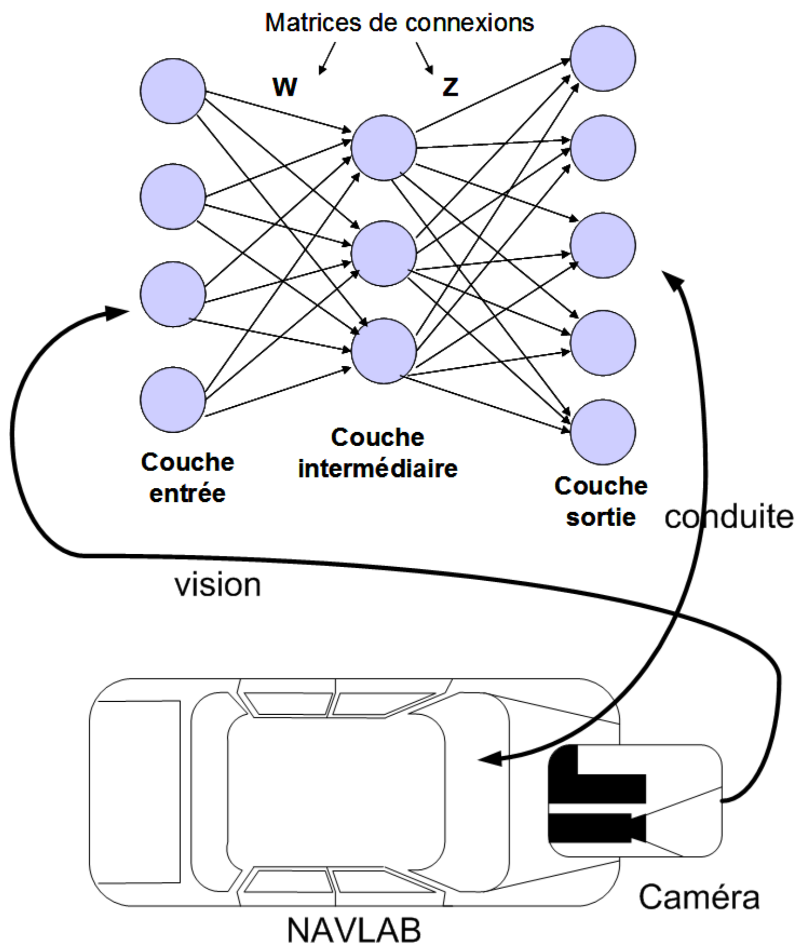
Neural Networks




Matrices de connexions
W
Z
Couche entrée
Couche intermédiaire
Couche sortie
conduite
vision
Caméra
NAVLAB

Plan

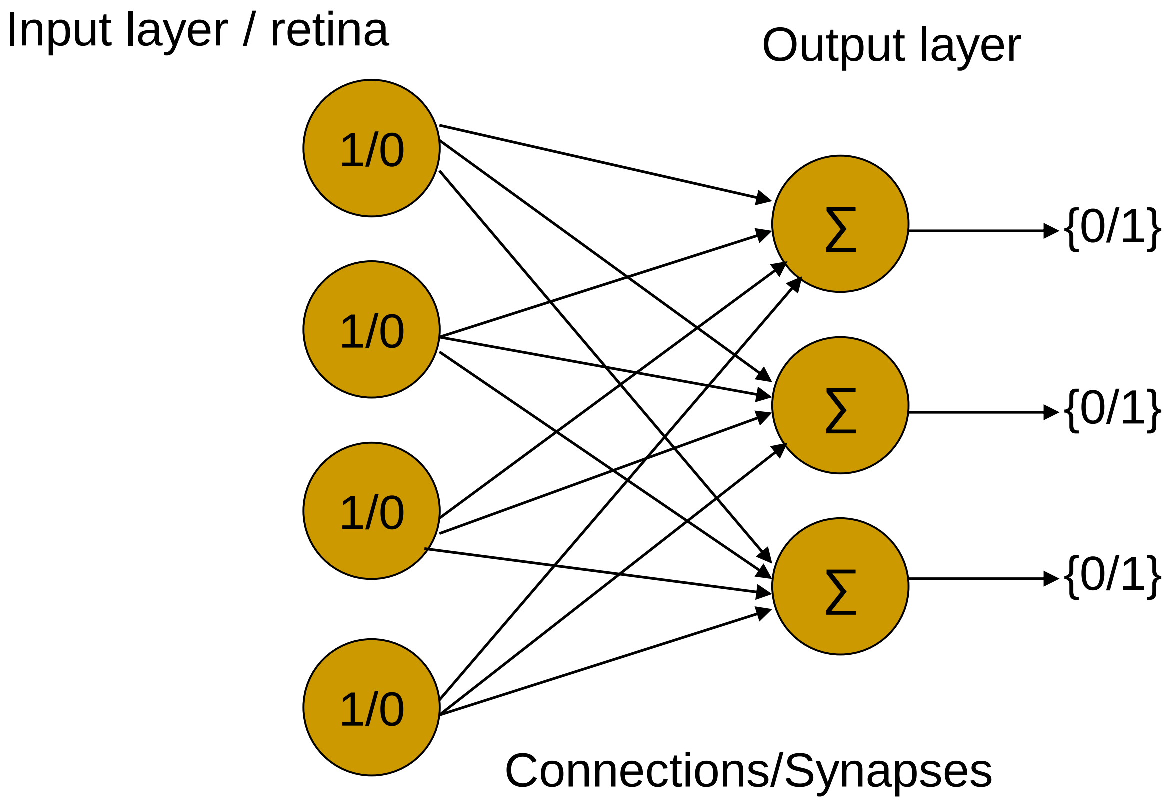
- Perceptron
  - Linear discriminant
- Associative memories
  - Hopfield networks
  - Chaotic networks
- Multilayer perceptron
  - Backpropagation

Perceptron

- Historically, the first neural net
- Inspired by human brain
- Proposed
  - By Rosenblatt
  - Between 1957 et 1961
- The brain was appearing as the best computer
- Goal: associated input patterns to recognition outputs
- Akin to a linear discriminant

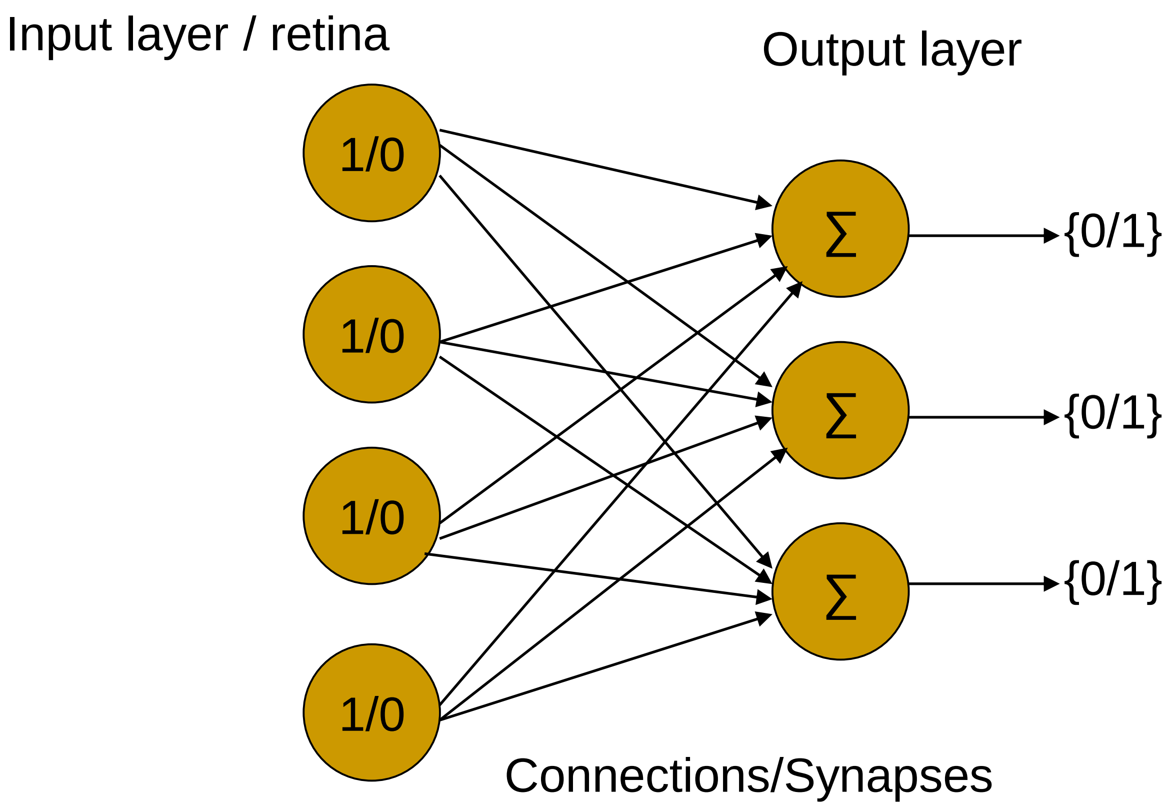
Perceptron

- Constitution



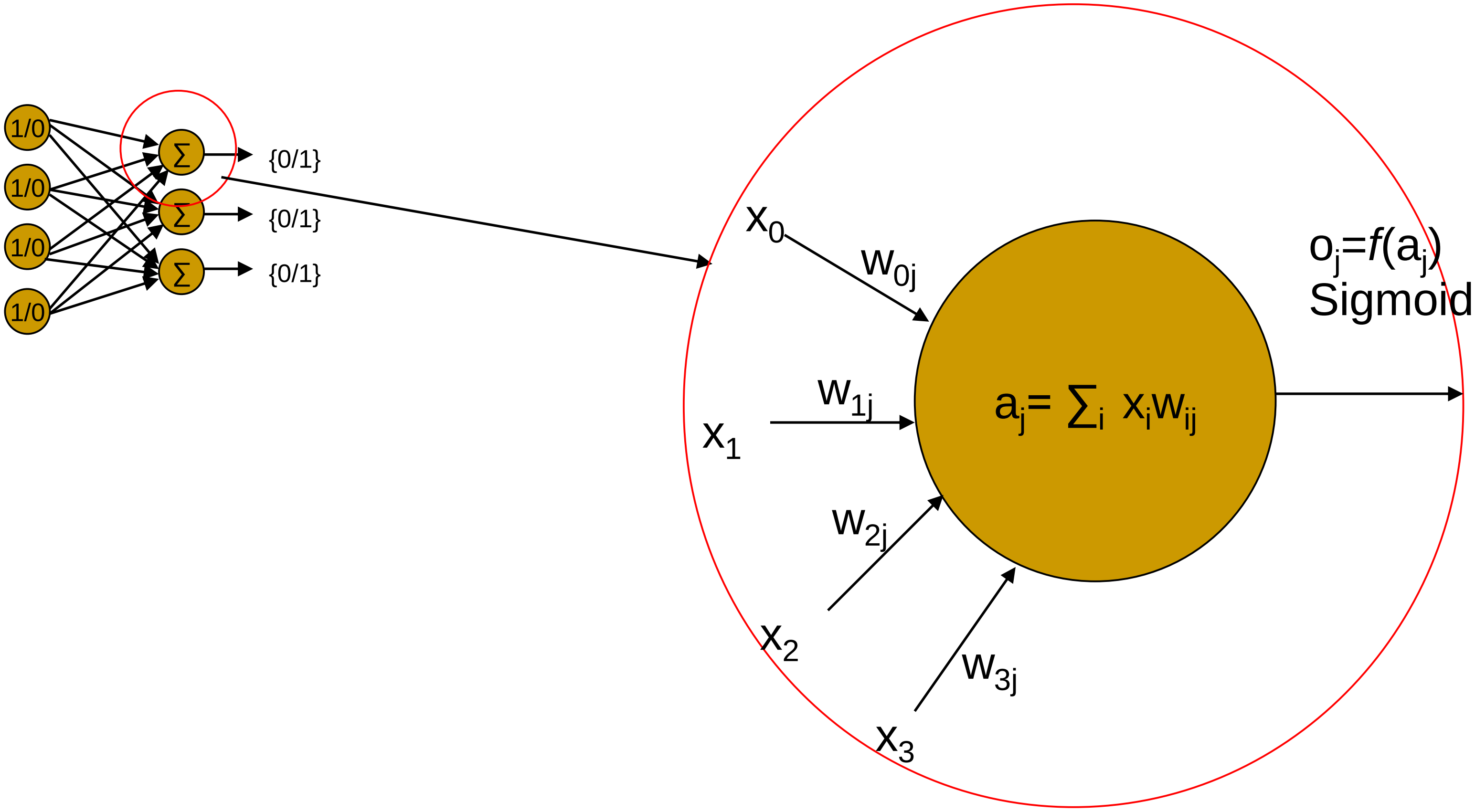
Perceptron

- Constitution



Perceptron

- Constitution



Perceptron

- Constitution

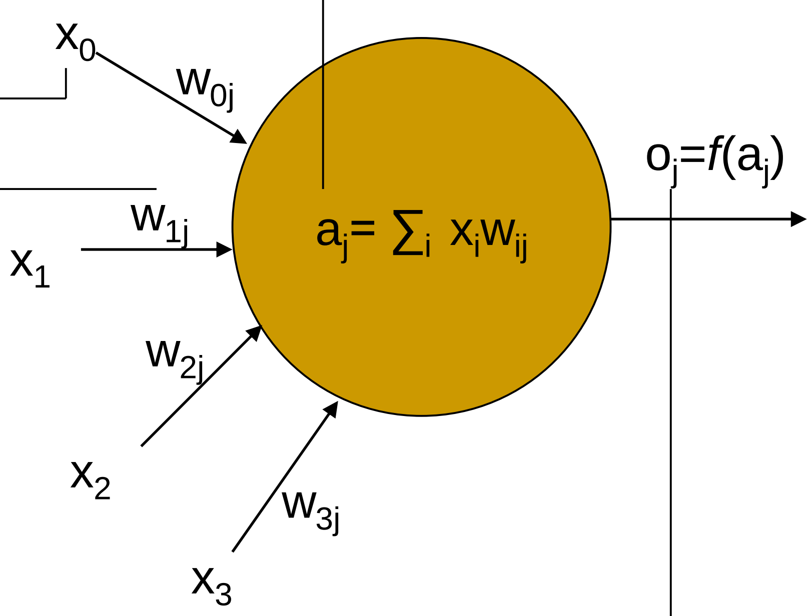
a_j : activation of the j output neuron

x_i : activation of the i input neuron

$w_{i,j}$: connection parameter between input neuron i and output neuron j

o_j : decision rule

$o_j = 0$ for $a_j <= \theta_j$, 1 for $a_j > \theta_j$



Perceptron

- Need an associated learning
  - Learning is supervised
    - Based on a couple input pattern and desired output
  - If the activation of output neuron is OK => nothing happens
  - Otherwise – inspired by neurophysiological data
    - If it is activated : decrease the value of the connection
    - If it is unactivated : increase the value of the connection
  - Iterated until the output neurons reach the desired value

Perceptron

- Supervised learning
  - How to decrease or increase the connections ?
  - Learning rule of Widrow-Hoff
  - Closed to Hebbian learning

$$\boxed{w_{i,j}^{(t+1)}} = \boxed{w_{i,j}^{(t)}} + \boxed{n(t_j - o_j)x_i} = w_{i,j}^{(t)} + \Delta w_{i,j}$$

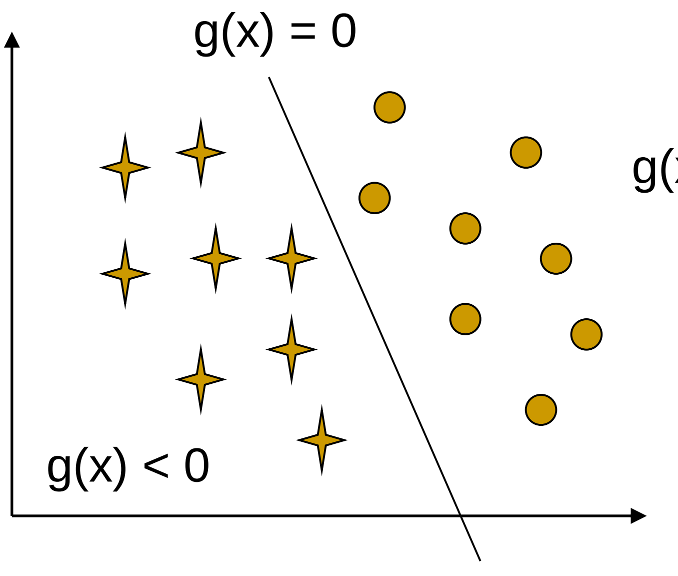
Desired value of output neuron j

Learning rate

Theory of linear discriminant

Compute:

$g(x) = W^T x + W_o$



And:

Choose:

class 1 if $g(x) > 0$

class 2 otherwise

But how to find W on the basis of the data ?

Gradient descent:

$$\Delta W_i = -\eta \frac{\partial E}{\partial W_i}, \forall i$$

In general a sigmoid is used for the statistical interpretation: (0,1)

$$Y = 1/1 + \exp\left[-g(x)\right]$$

Easy to derive = Y(1-Y)

Class 1 if Y > 0.5 and 2 otherwise

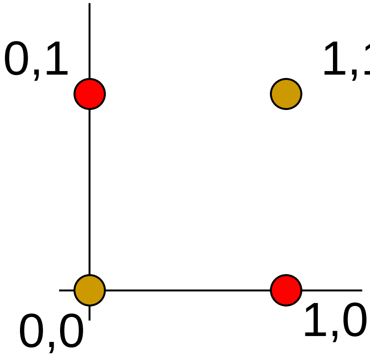
The error could be least square: $(Y - Y_d)^2$

Or maximum likelihood: $-\sum Y_d \log Y + (1 - Y_d)\log(1 - Y)$

But at the end, you got the learning rule: $\Delta W = \eta \sum (Yd - Y) X_j$

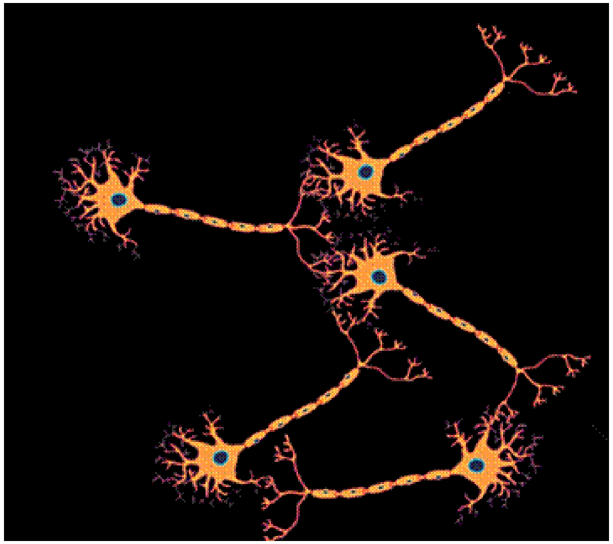
Perceptron limitations

- Limitations
  - Not always easy to learn
  - But above all, cannot separate not linearly separable data
- Why so ?
  - The XOR kills NN researches
    for 20 years
    (Minsky and Papert were responsable)



- Consequence
  - We had to wait for the magical hidden layer
  - And for backpropagation

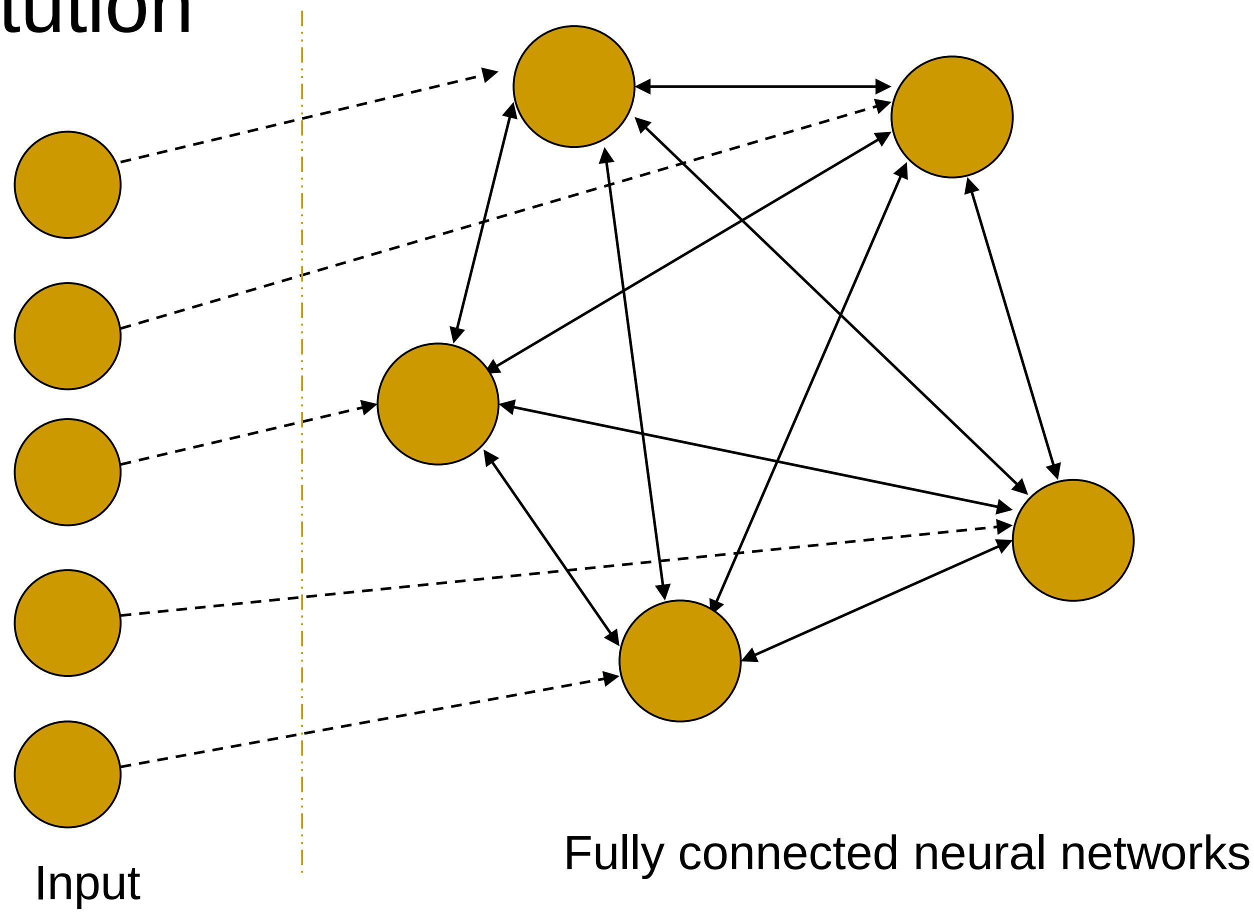
Associative memories



- Around 1970
- Two types
  - Hetero-associative
  - And auto-associative
- We will treat here only auto-associative
- Make an interesting connections between neurosciences and physics of complex systems
- John Hopfield

Auto-associative memories

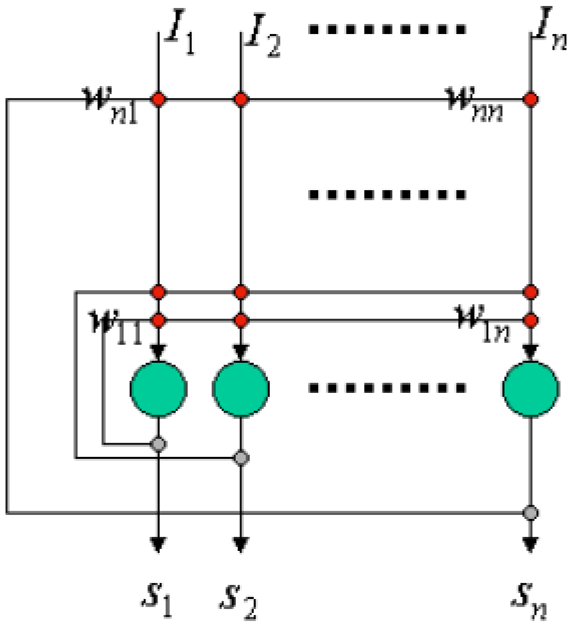
- Constitution



Hopfield networks
recurrent networks and memories

$$h_i(t) = \sum_{j=1}^{N} w_{ij} s_j(t) + \theta_i$$

$$s_i(t+1) = \mathrm{sgn}\left[h_i(t)\right]$$



1. Statistical analysis using spin models
   - According to weight constrains (weights have to be symmetric), the dynamics iterates to stable patterns: fixed point attractors, i.e. '**memories**'

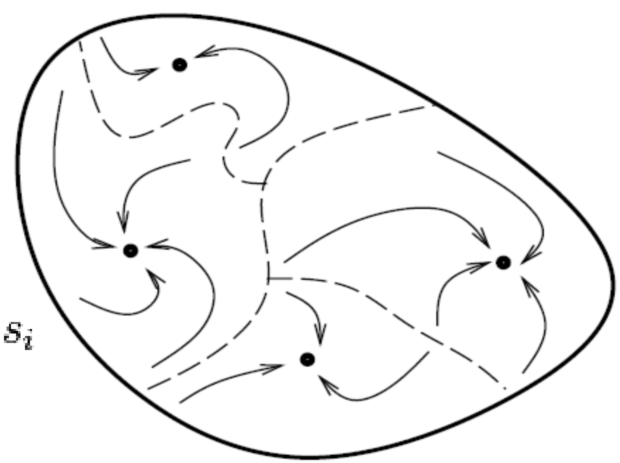
2. Hebbian learning of the desired attractors

$$\{\xi^{\mu} = (\xi_1^{\mu}, \xi_2^{\mu} \ldots \xi_N^{\mu});\ 1 \leq \mu \leq M\}$$

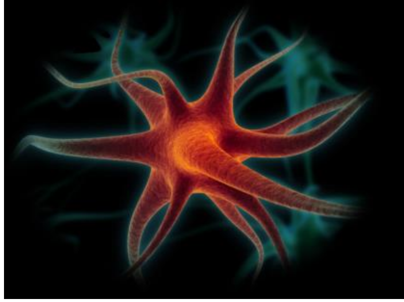
$$w_{ij} = \frac{1}{N} \sum_{p=1}^{M} \xi_i^p \xi_j^p$$

3. Minimize the Energy Function

$$E = -\sum_{i<j} w_{ij}\, s_i\, s_j + \sum_i \theta_i\, s_i$$

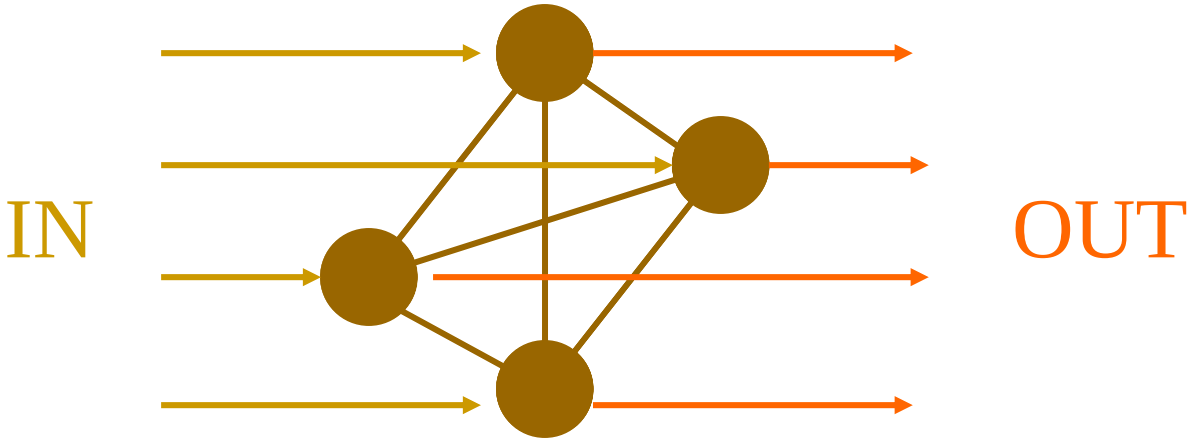


Associative memories



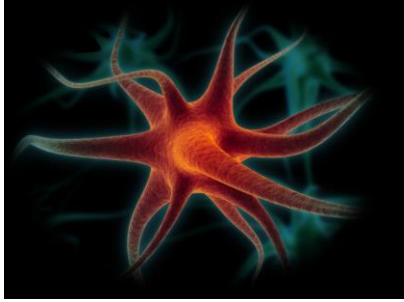
Hopfield -> DEMO

- Fully connected graphs
- Input layer = Output layer = Networks
- The connexions have to be symmetric



- It is again an hebbian learning rule

Associative memories

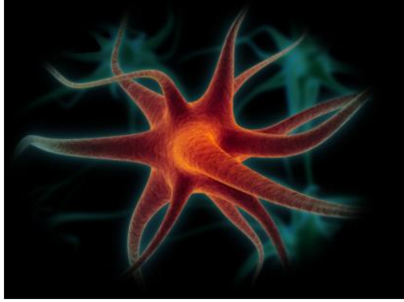


Hopfield

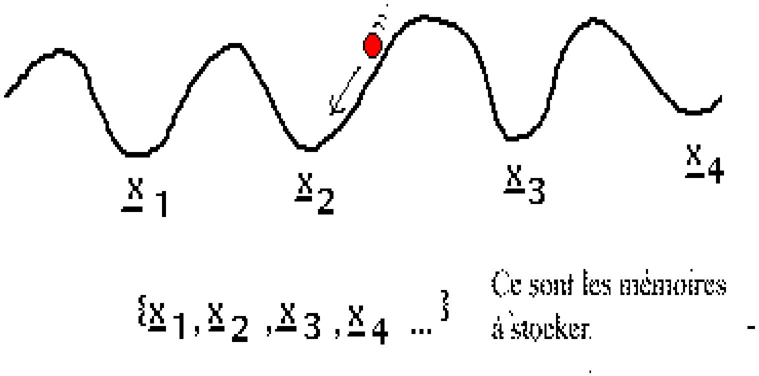
- The newtork becomes a dynamical machine
- It has been shown to converge into a fixed point
- This fixed point is a minimal of a Lyapunov energy

- These fixed point are used for storing «patterns »
- Discrete time and asynchronous updating
  - input in $\{-1,1\}$
  - $x_i \rightarrow \mathrm{sign}(\sum_j w_{ij} x_j)$

Mémoires associatives

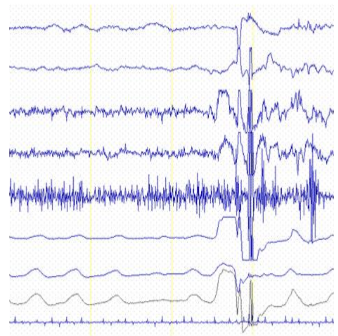


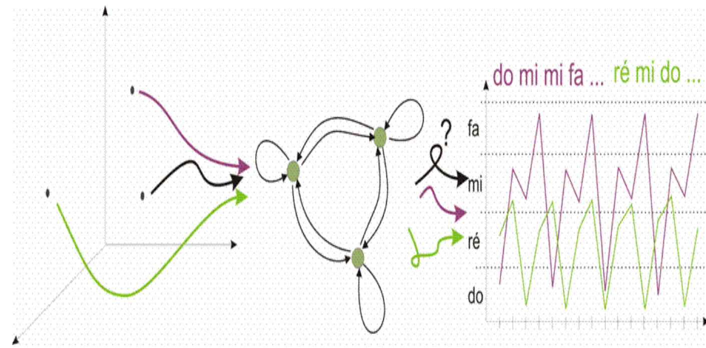
Hopfield

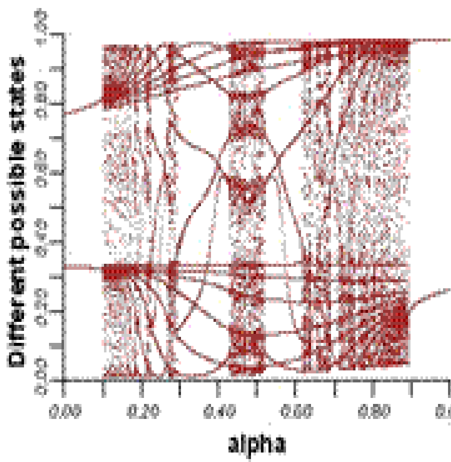


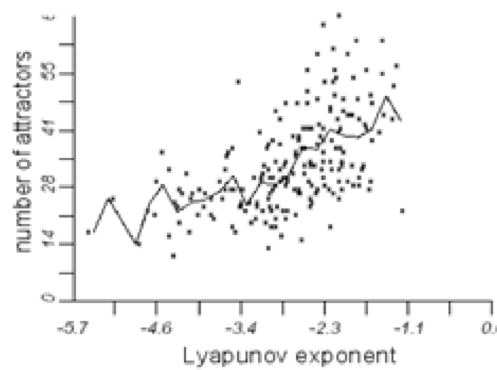
- The learning is done by Hebbian learning
- Over all patterns to learn:
  - $$\Delta W_{ij} = \sum_{patterns} X_i^p X_j^p$$

My researches: Chaotic encoding of memories in brain



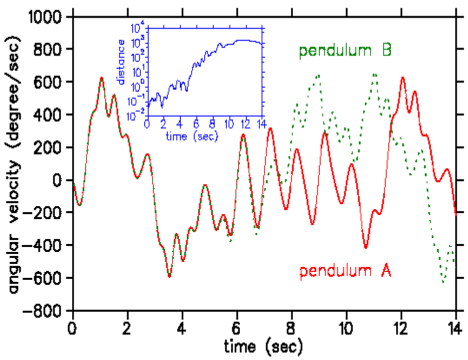






What is “chaos”?

Chaos – is a *aperiodic* long-time behavior arising in a *deterministic* dynamical system that exhibits a *sensitive dependence on initial conditions*.



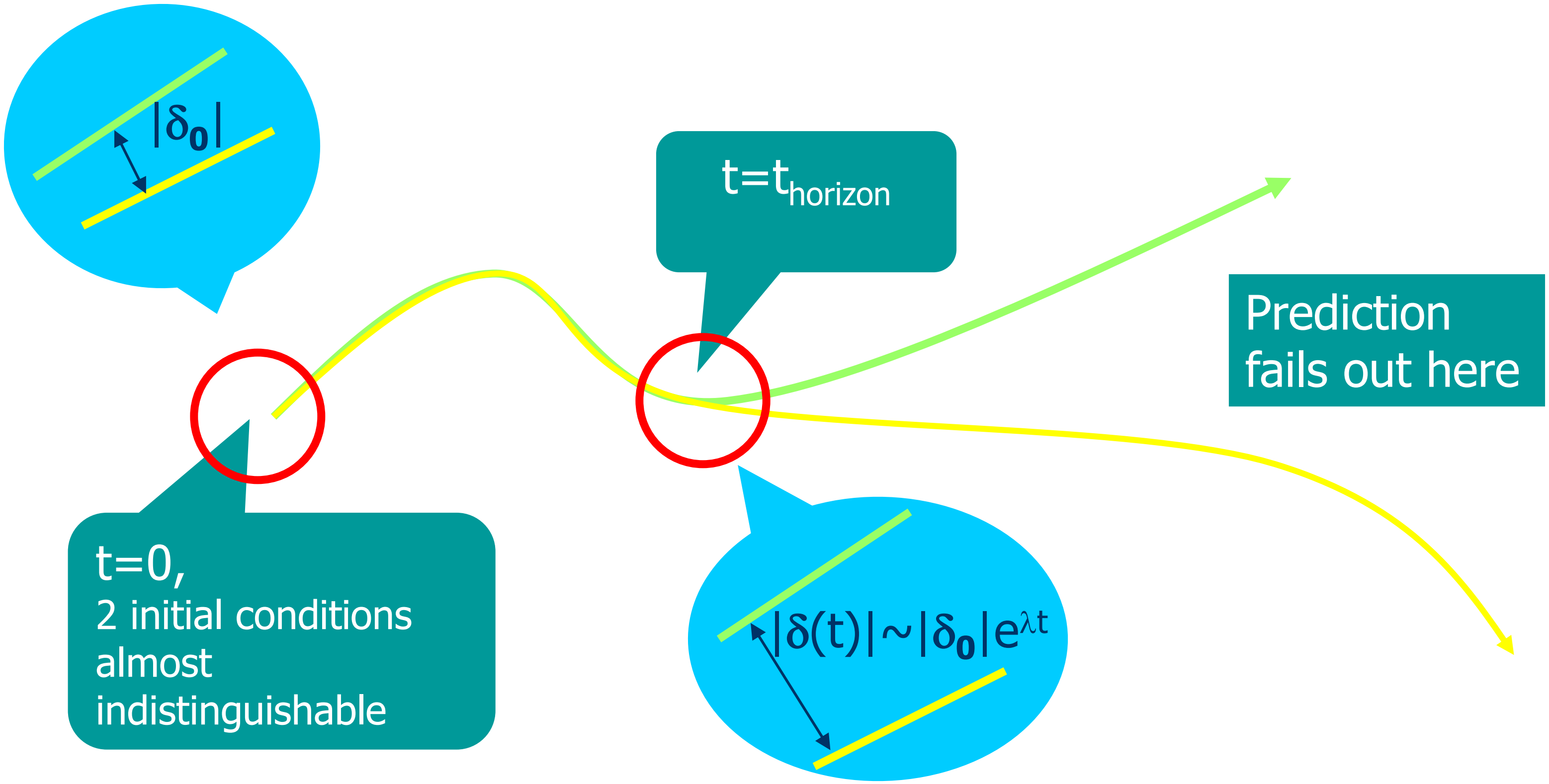
The nearby trajectories separate exponentially fast

Lyapunov Exponent > 0

Trajectories which do not settle down to fixed points, periodic orbits or quasiperiodic orbits as $t \to \infty$

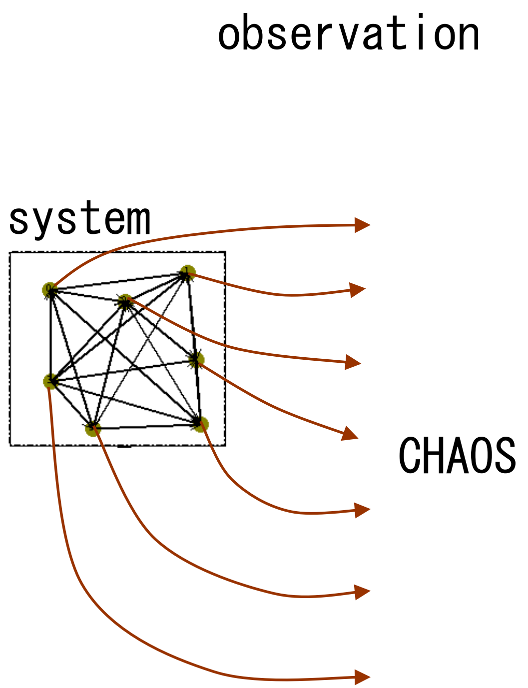
The system has no random or noisy inputs or parameters – the irregular behavior arises from system’s nonliniarity

Demonstration

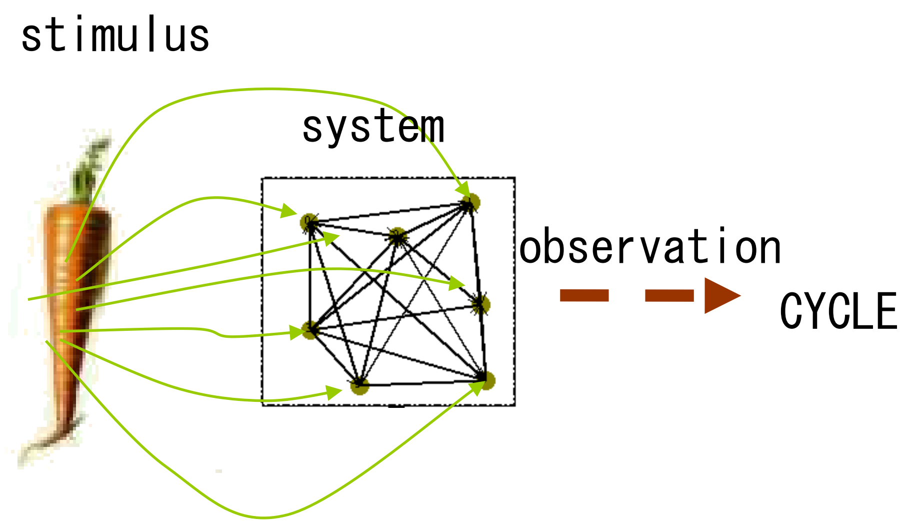


- No matter how hard we work to reduce measurement error, we cannot predict longer than a few multiples of $1/\lambda$.

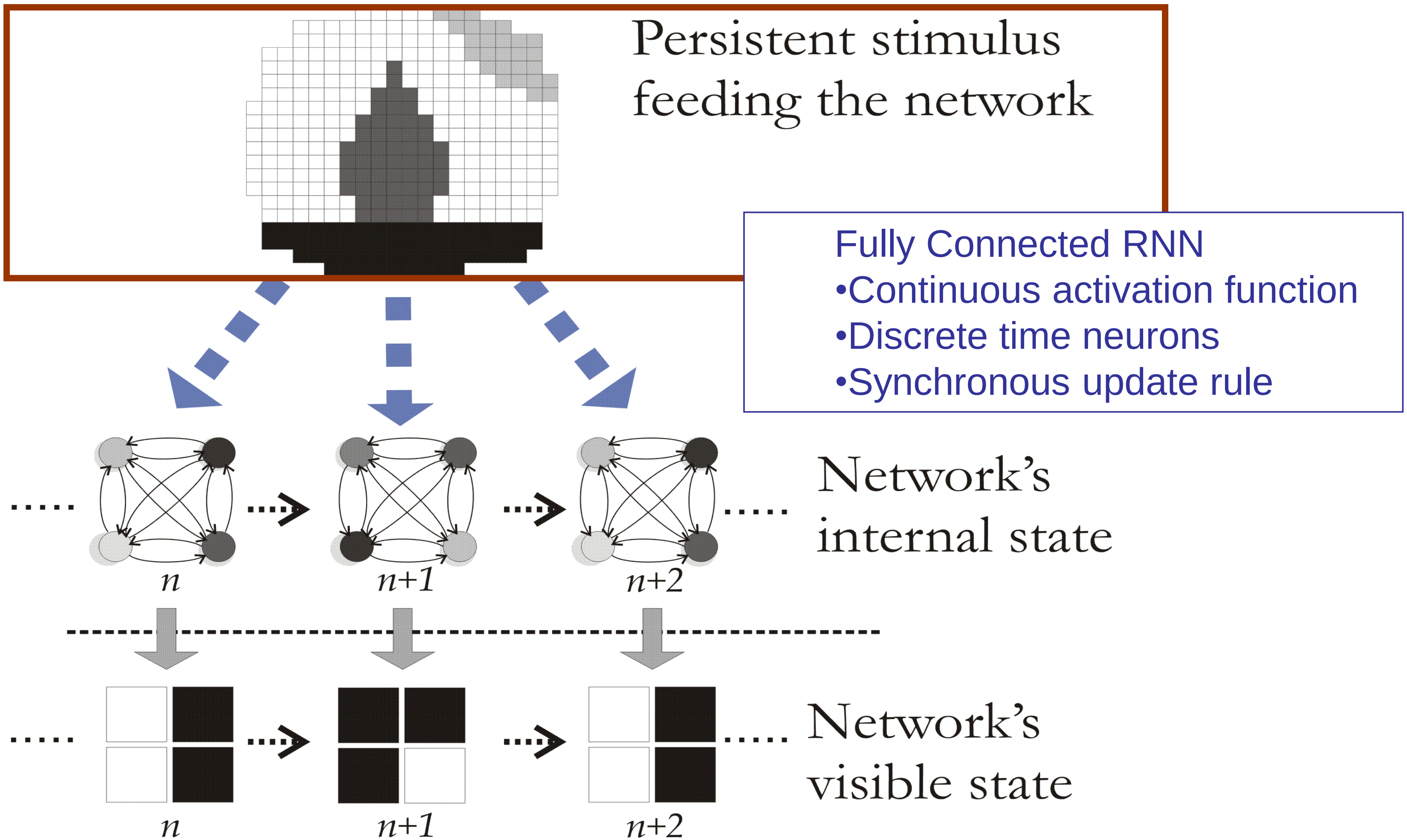
In absence of information



Storing information

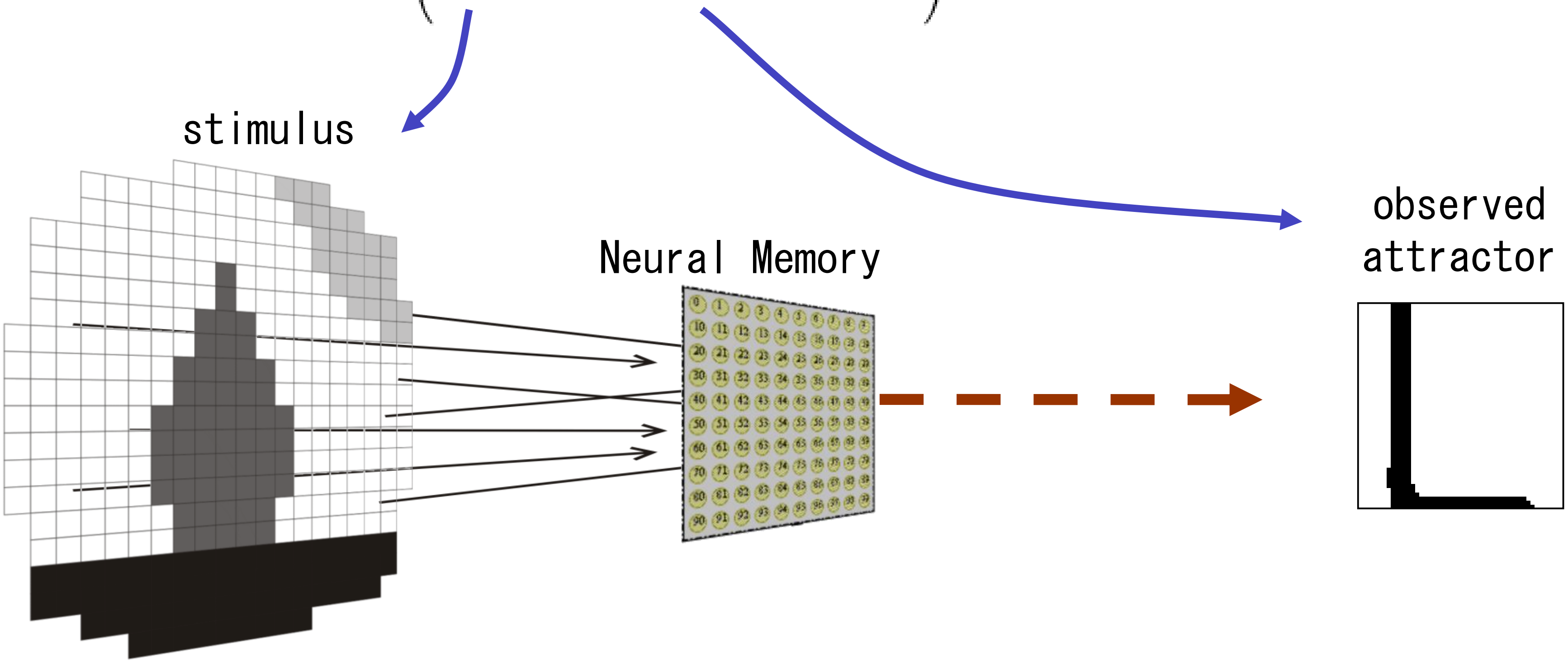


The model

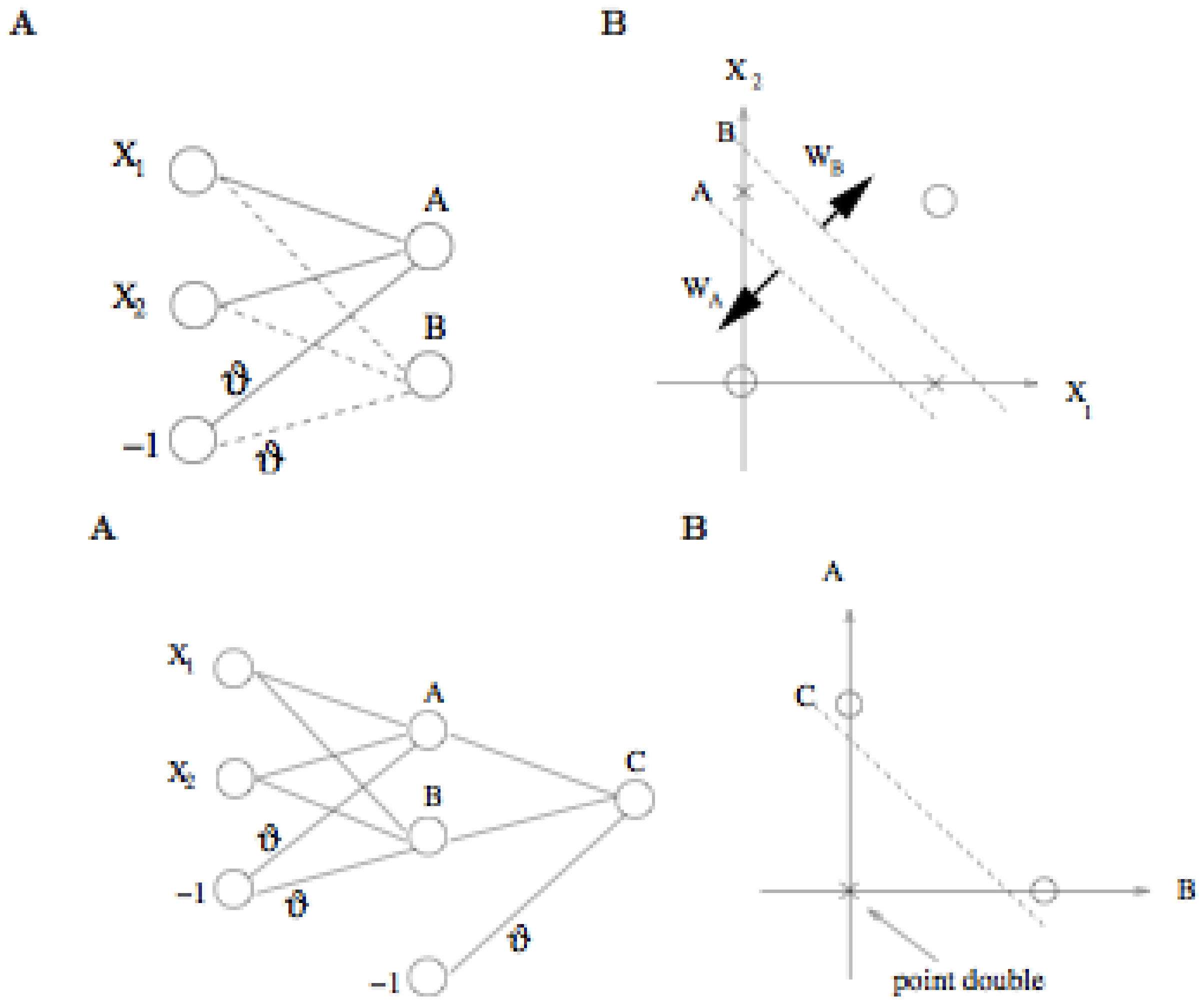

Persistent stimulus feeding the network
Fully Connected RNN
•Continuous activation function
•Discrete time neurons
•Synchronous update rule
Network's internal state
n
n+1
n+2
Network's visible state
n
n+1
n+2

The "out-supervised" learning task

$$\mathcal{D}^{\mu} = \left(\chi^{\mu}, (\boldsymbol{\varsigma}^{\boldsymbol{\mu,1}}, \varsigma^{\mu,2}, \varsigma^{\mu,3}, \varsigma^{\mu,4}, \varsigma^{\mu,5})\right) \qquad \mu = 1, \ldots, q$$

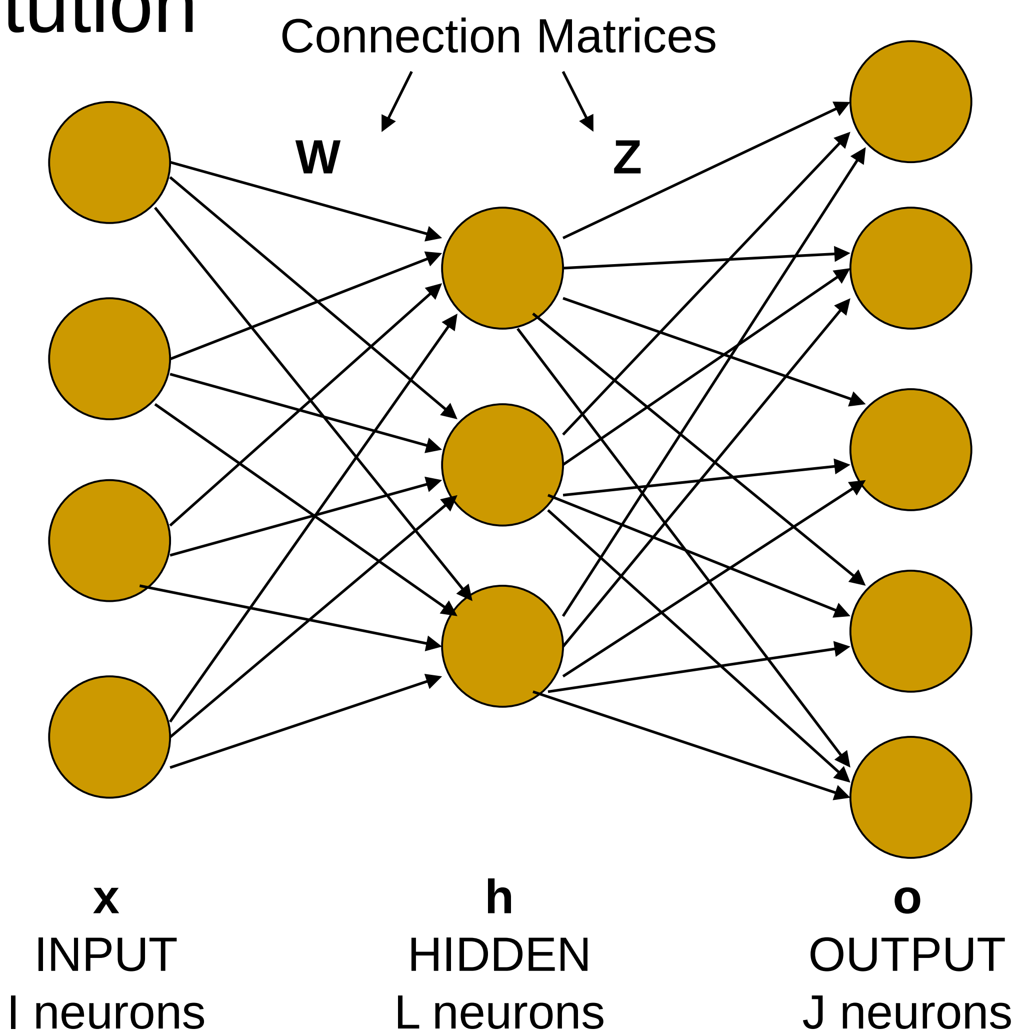


Multi-Layers Perceptron
the XOR problem



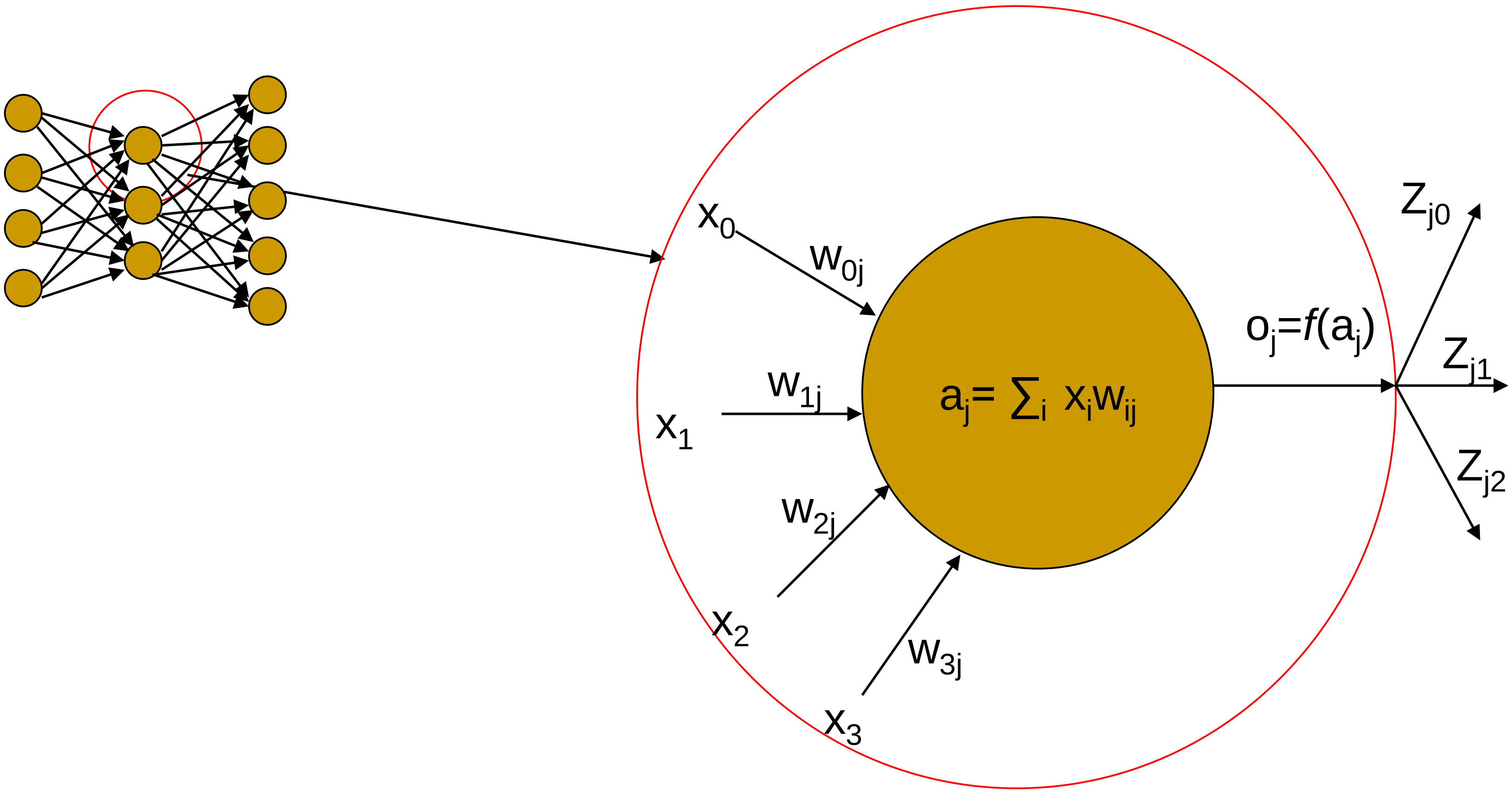
Multilayer perceptron

- Constitution



Multilayer Perceptron

- Constitution



Error backpropagation

- Learning algorithm
- How it proceeds :
  - Inject an input
  - Get the output
  - Compute the error with respect to the desired output
  - Propagate this error back from the output layer to the input layer of the network
  - Just a consequence of the chaining derivative of the gradient descent

Backpropagation

- Select a derivable transfert function
  - Classicaly used : The logistics

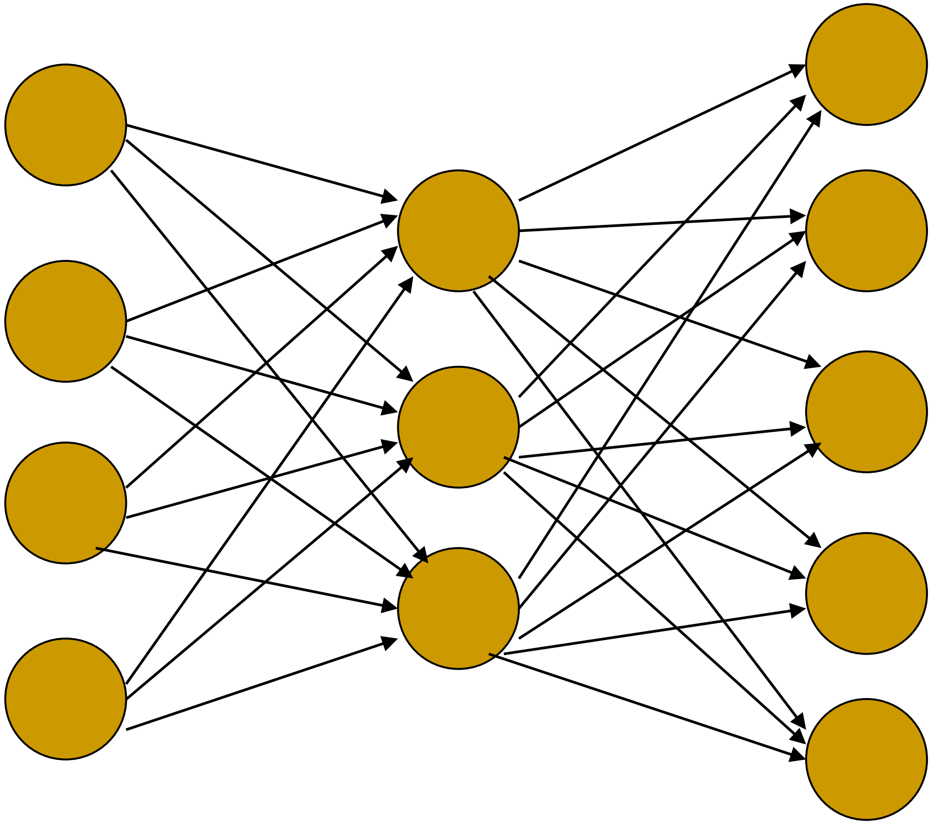
$$f(x) = \frac{1}{1+e^{-x}}$$

  - And its derivative

$$f'(x) = f(x)[1 - f(x)]$$

Backpropagation

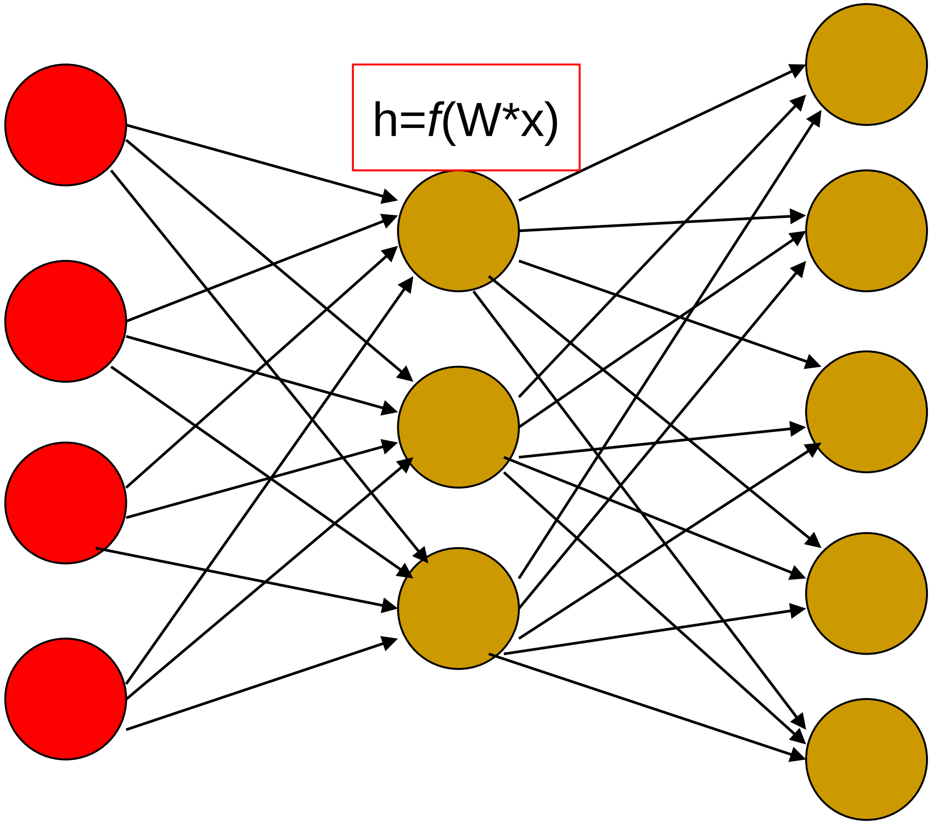
- The algorithm



1. Inject an entry

Backpropagation

- Algorithm

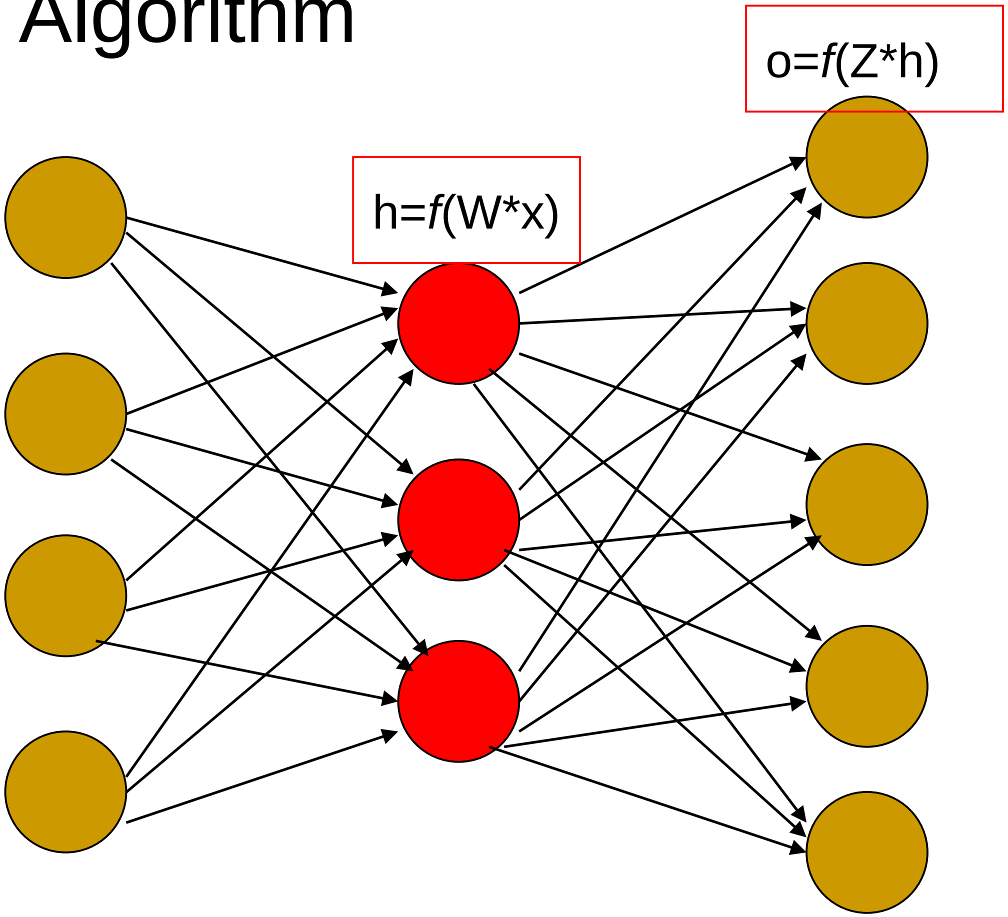


1. Inject an entry

2. Compute the intermediate h

Backpropagation

- Algorithm



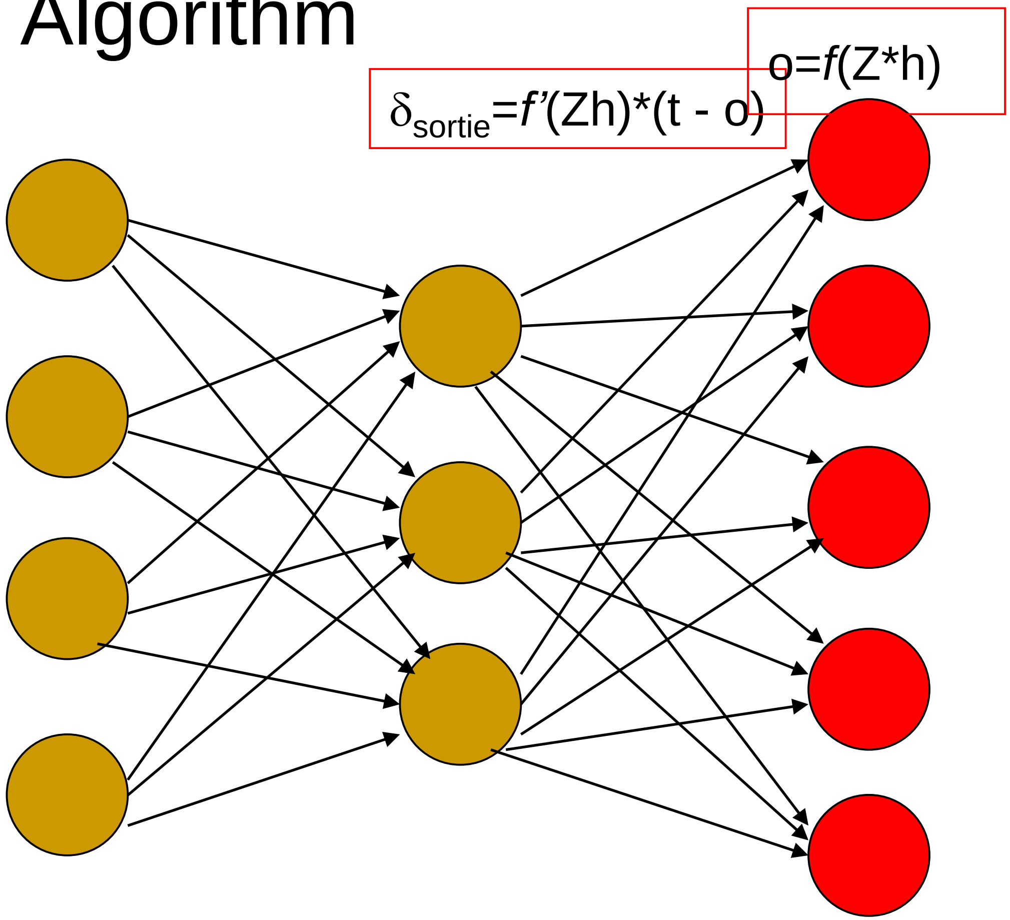
1. Inject an entry

2. Compute the intermediate h

3. Compute the output o

Backpropagation

- Algorithm

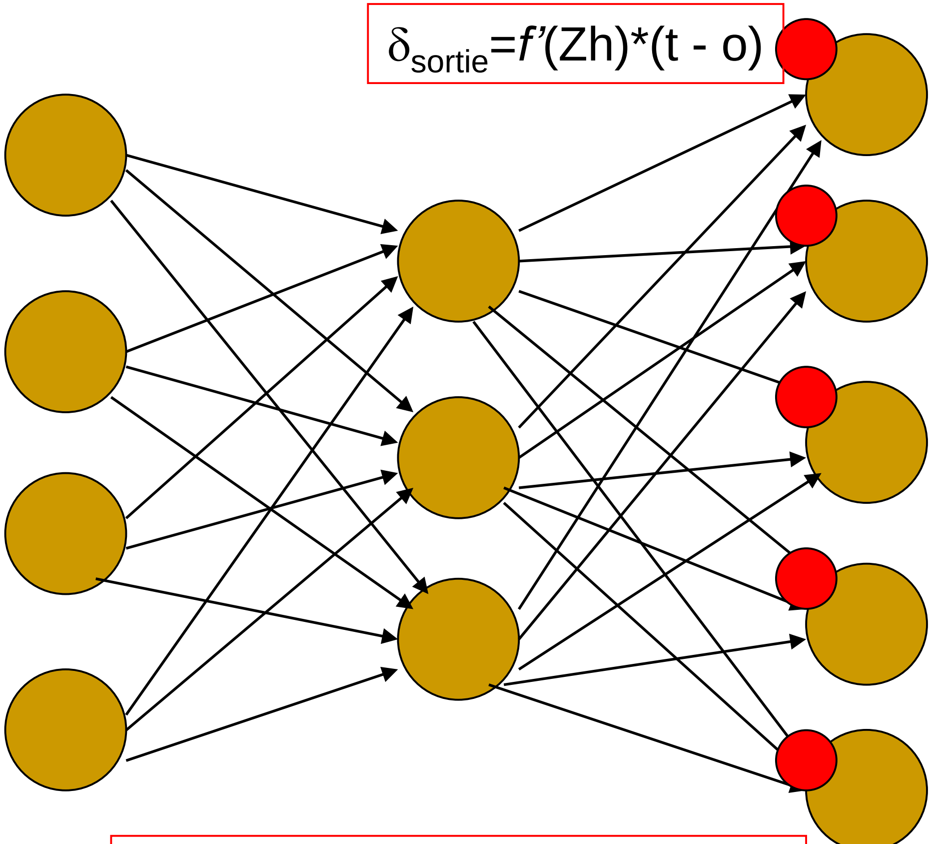


1. Inject an entry
2. Compute the intermediate h
3. Compute the output o
4. Compute the error output

Backpropagation

- Algorithm

$\delta_{sortie} = f'(Zh)*(t - o)$



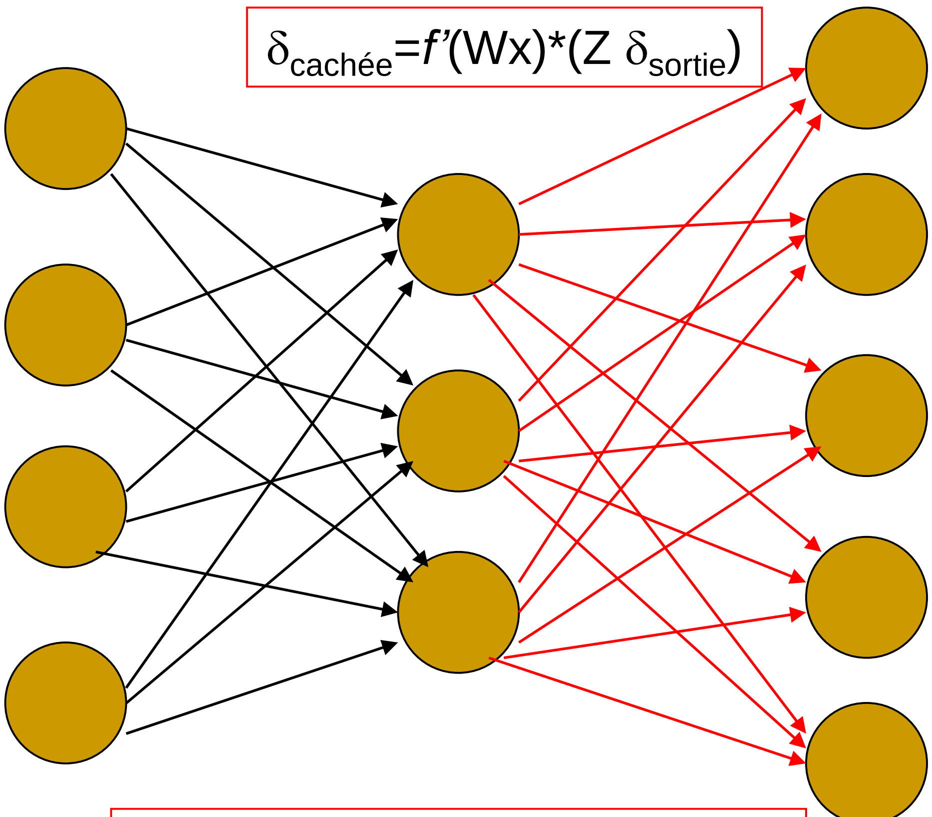
$Z^{(t+1)} = Z^{(t)} + n\ \delta_{sortie}\ h = Z^{(t)} + \Delta^{(t)} Z$

1. Inject an entry
2. Compute the intermediate h
3. Compute the output o
4. Compute the error output
5. Adjust Z on the basis of the error

Backpropagation

- Algorithm

$\delta_{cachée}=f'(Wx)*(Z\ \delta_{sortie})$



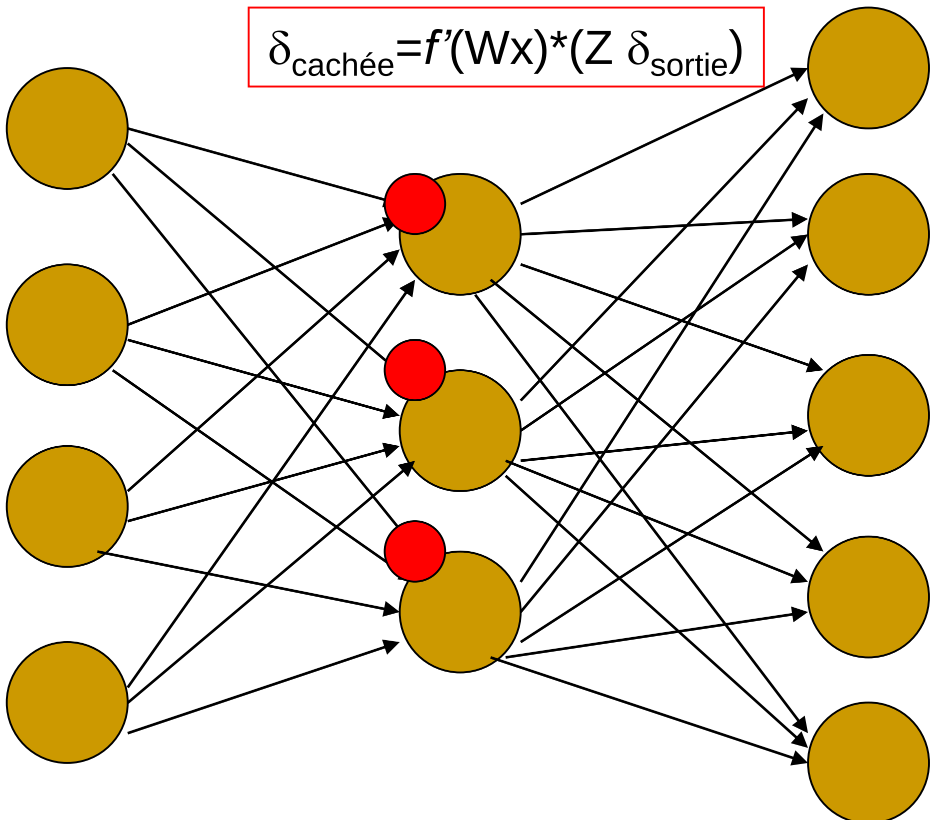
$Z^{(t+1)}=Z^{(t)}+n\ \delta_{sortie}\ h = Z^{(t)} + \Delta^{(t)}Z$

1. Inject an entry
2. Compute the intermediate h
3. Compute the output o
4. Compute the error output
5. Adjust Z on the basis of the error
6. Compute the error on the hidden layer

Backpropagation

- Algorithm

$\delta_{cachée}=f'(Wx)*(Z\ \delta_{sortie})$

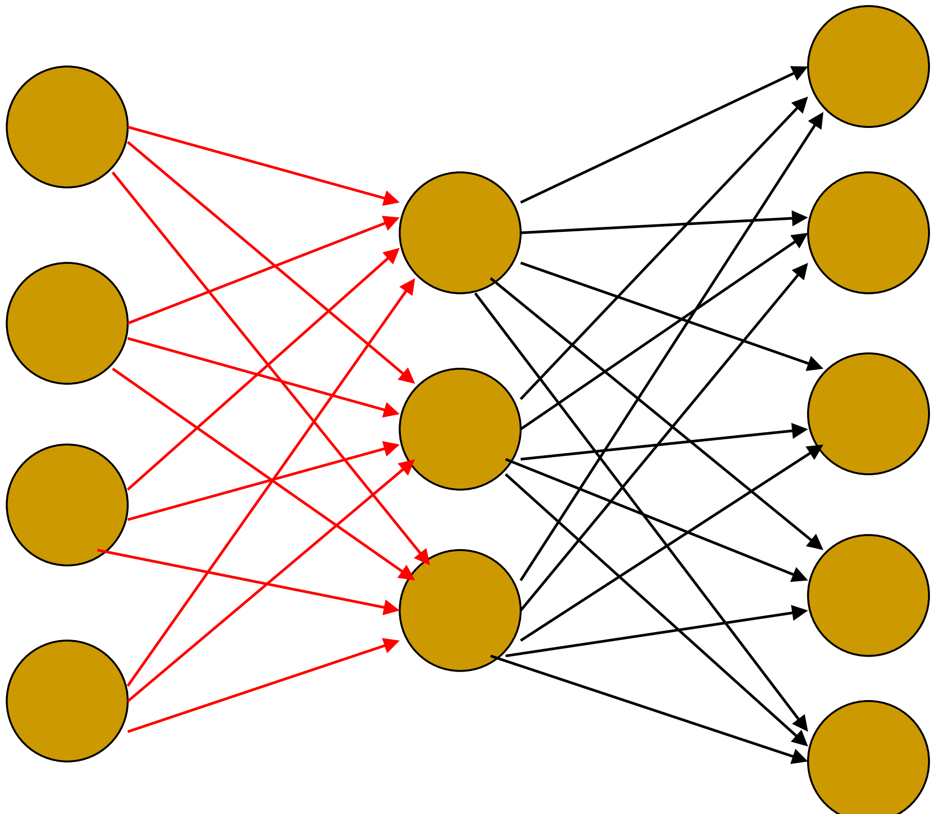


$W^{(t+1)}=W^{(t)}+n\ \delta_{cachée}\ x = W^{(t)} + \Delta^{(t)}W$

1. Inject an entry
2. Compute the intermediate h
3. Compute the output o
4. Compute the error output
5. Adjust Z on the basis of the error
6. Compute the error on the hidden layer
7. Adjust W on the basis of this error

Backpropagation

- Algorithm

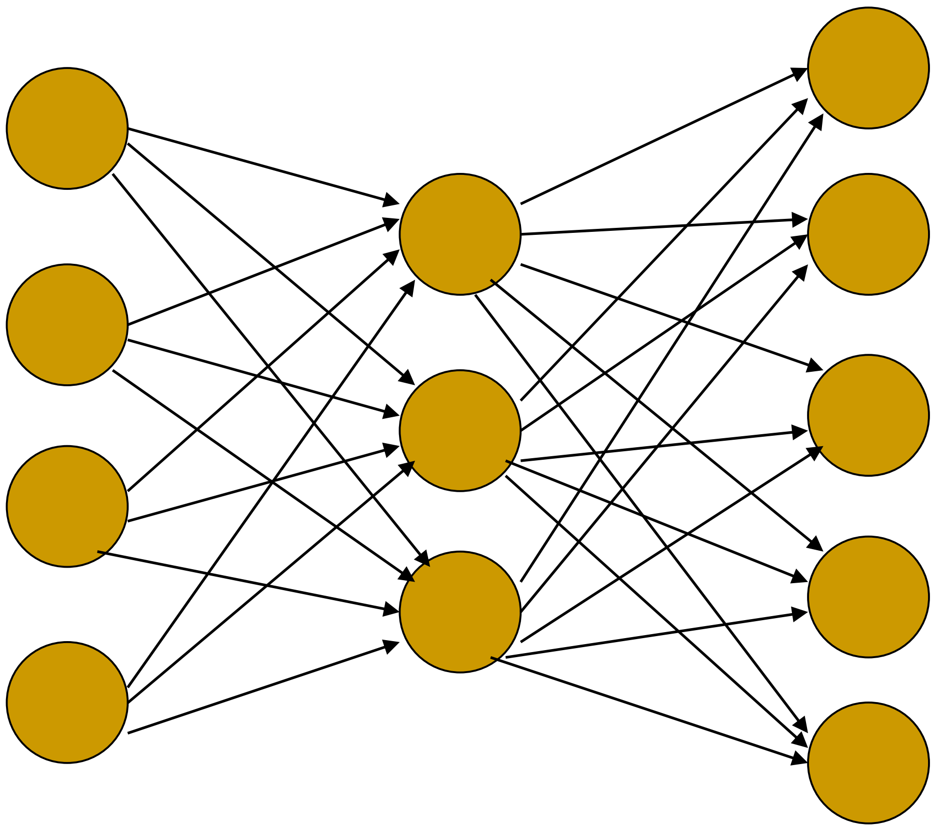


$W^{(t+1)}=W^{(t)}+n\ \delta_{cachée}\ x = W^{(t)} + \Delta^{(t)}W$

1. Inject an entry
2. Compute the intermediate h
3. Compute the output o
4. Compute the error output
5. Adjust Z on the basis of the error
6. Compute the error on the hidden layer
7. Adjust W on the basis of this error

Backpropagation

- Algorithm



1. Inject an entry

2. Compute the intermediate h

3. Compute the output o

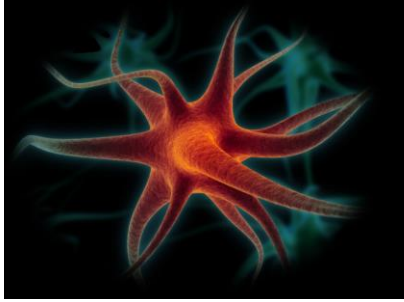
4. Compute the error output

5. Adjust Z on the basis of the error

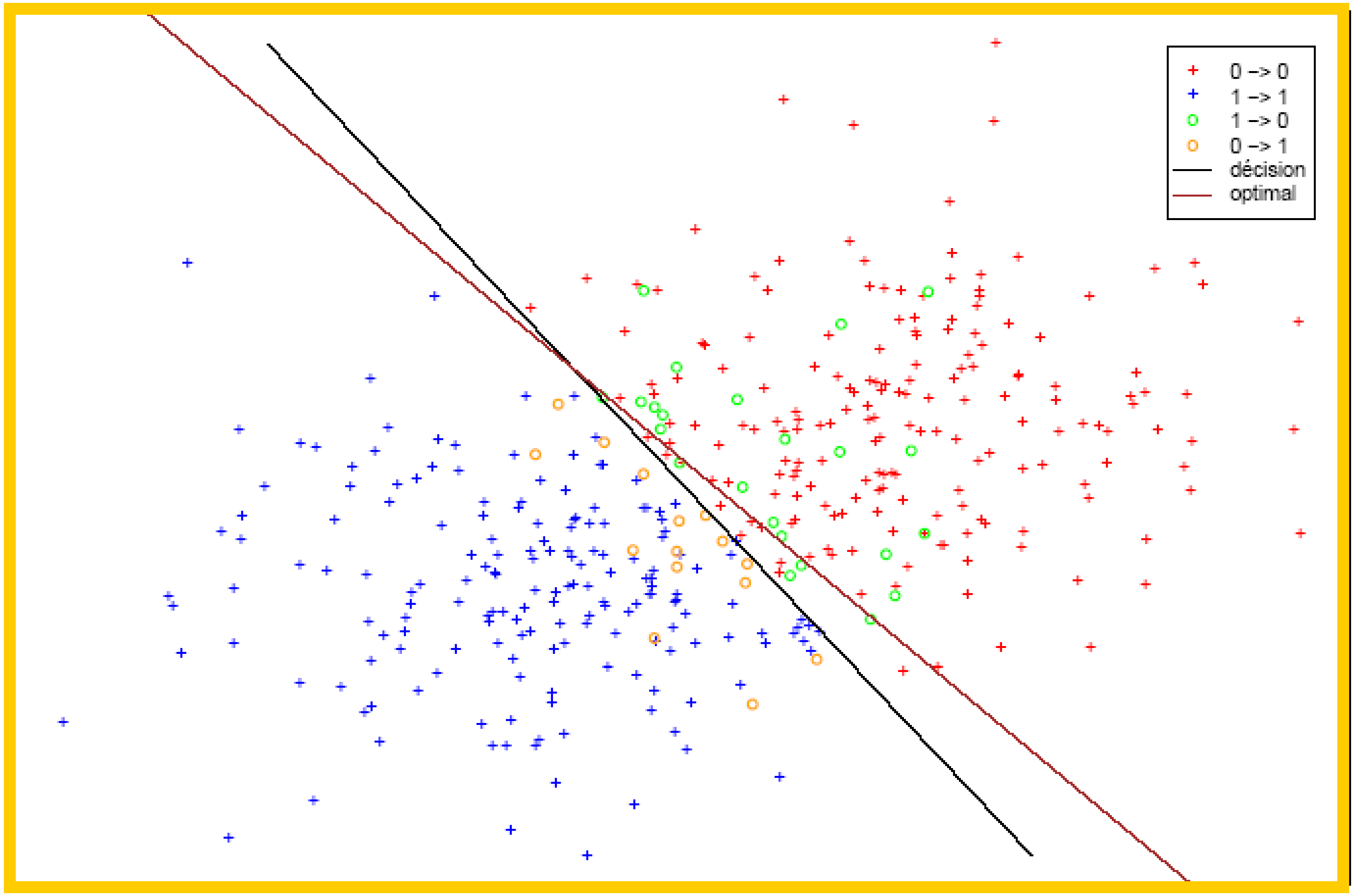
6. Compute the error on the hidden layer

7. Adjust W on the basis of this error

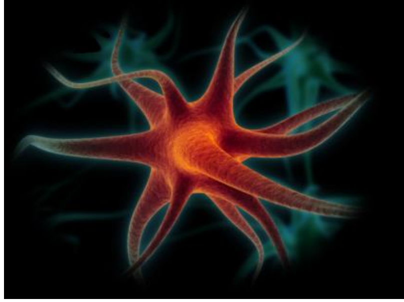
Neural network



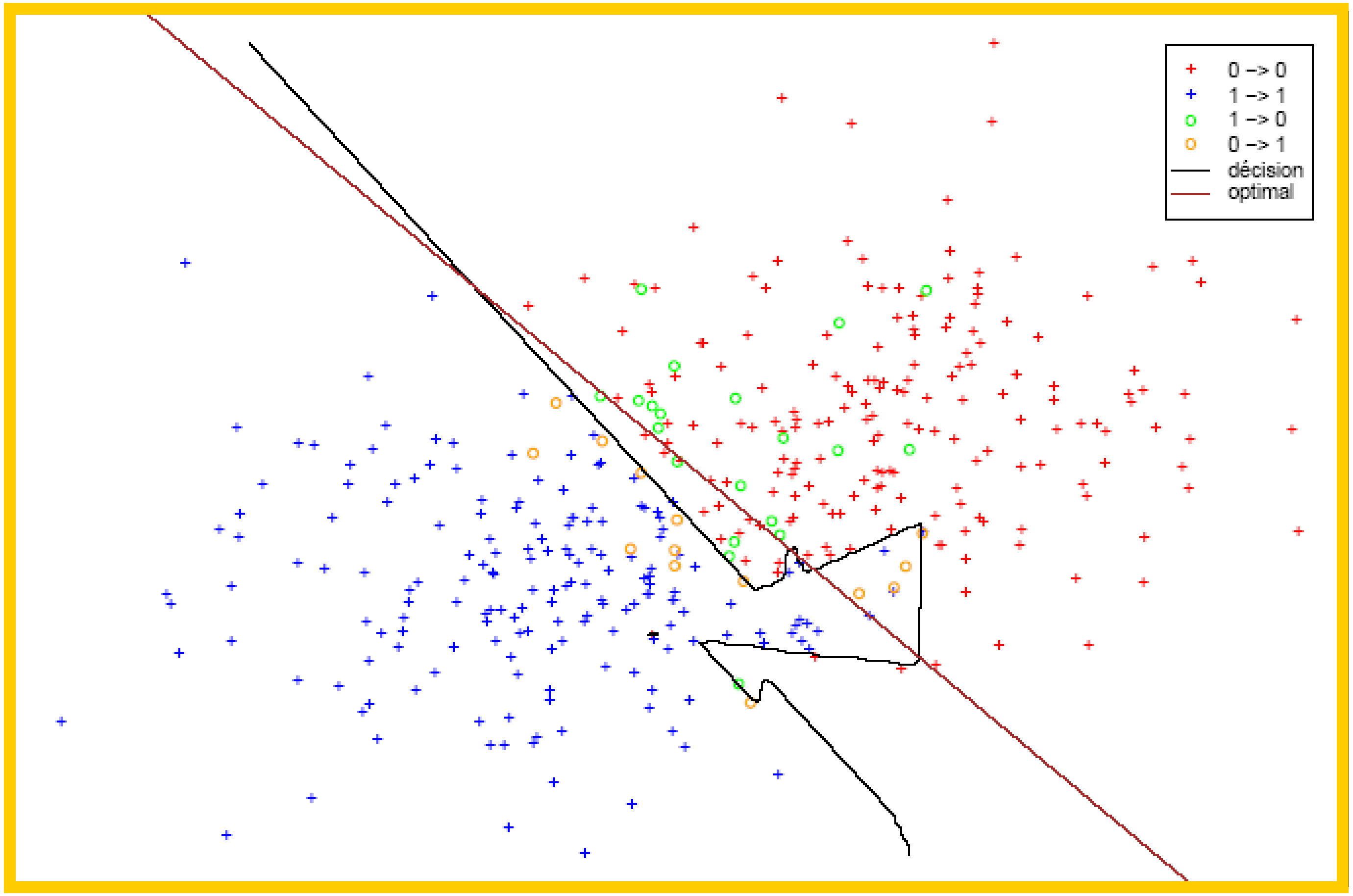
Simple linear discriminant



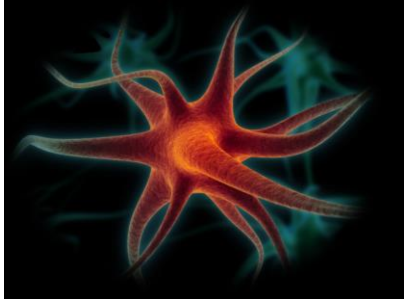
Neural networks



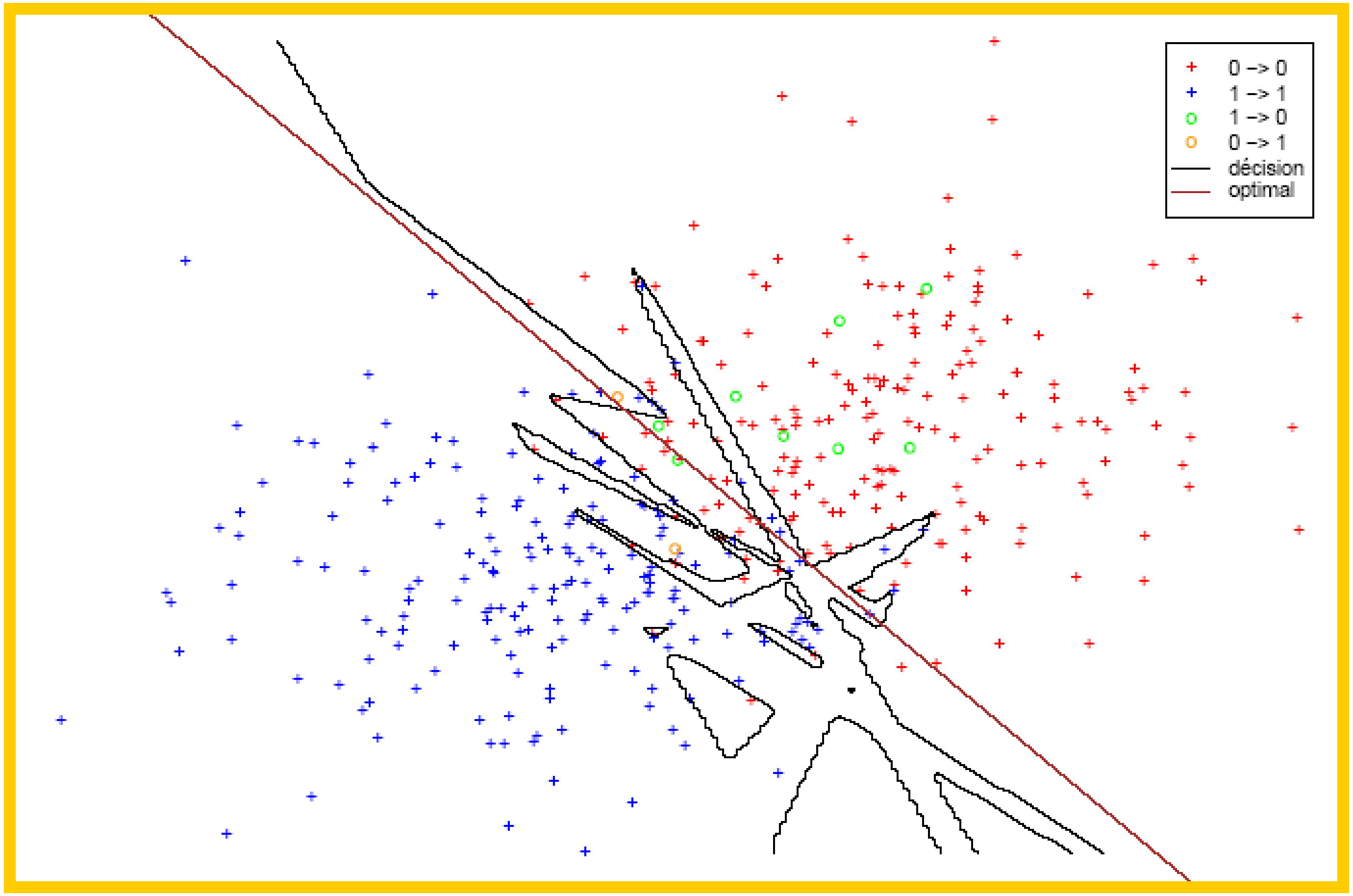
Few layers - Little learning

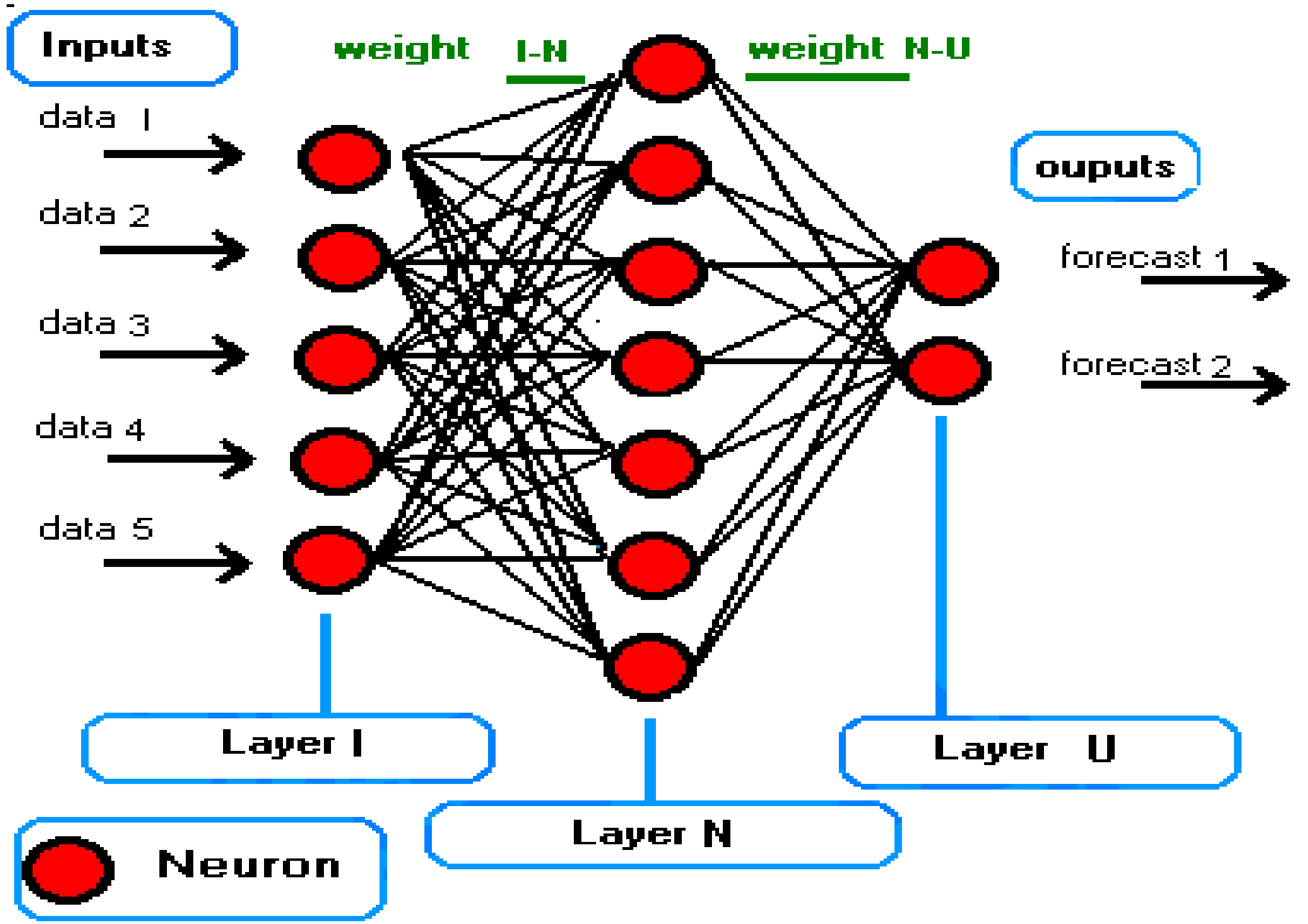


Neural networks



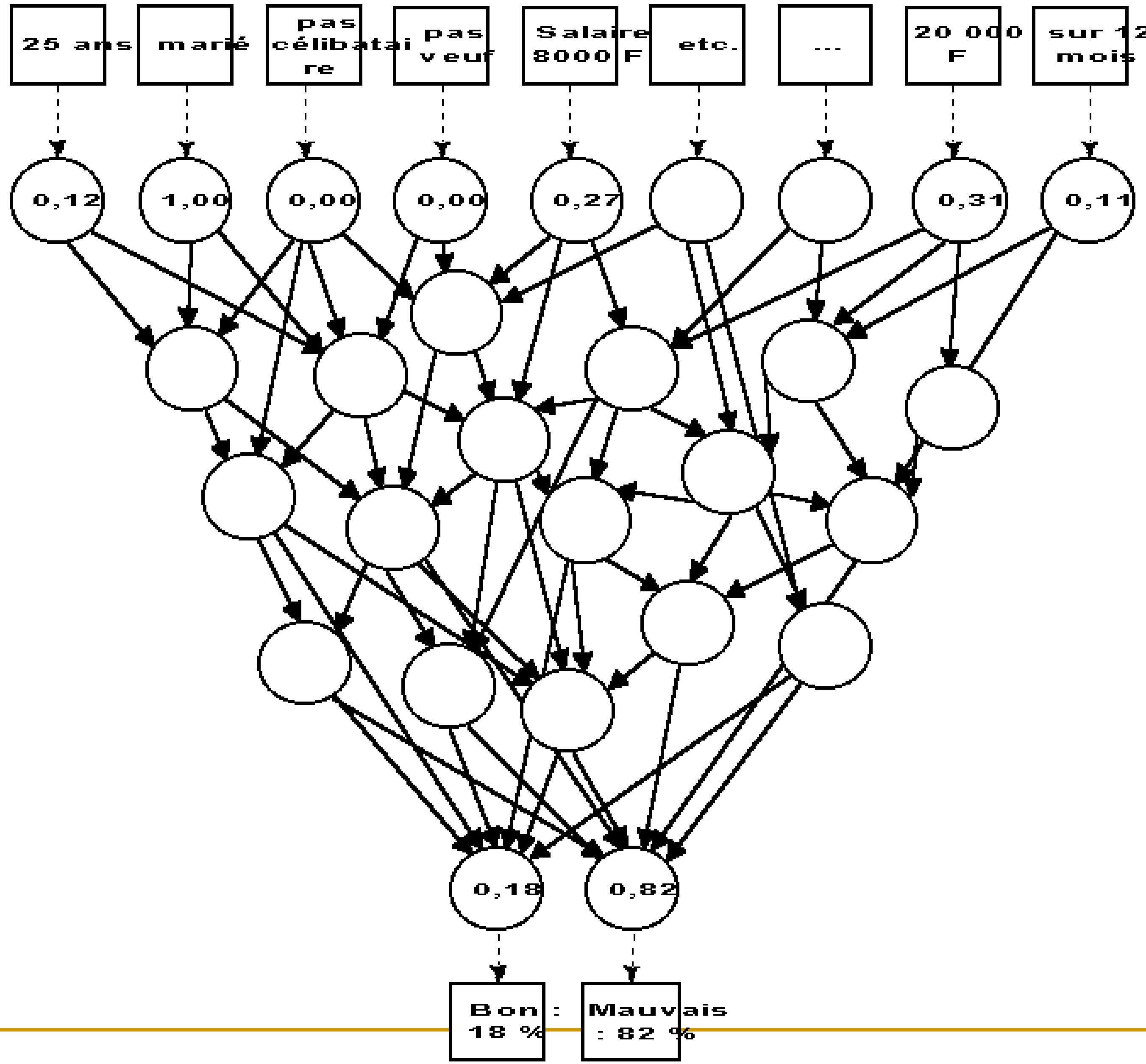
More layers - More learning





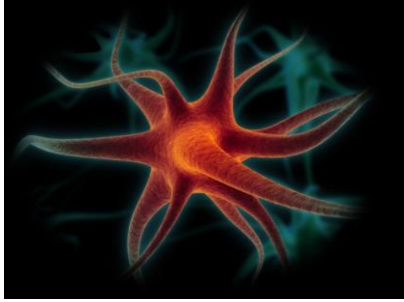
Inputs
weight I-N
weight N-U
data 1
data 2
data 3
data 4
data 5
ouputs
forecast 1
forecast 2
Layer I
Layer N
Layer U
Neuron

description de dossier de prêt



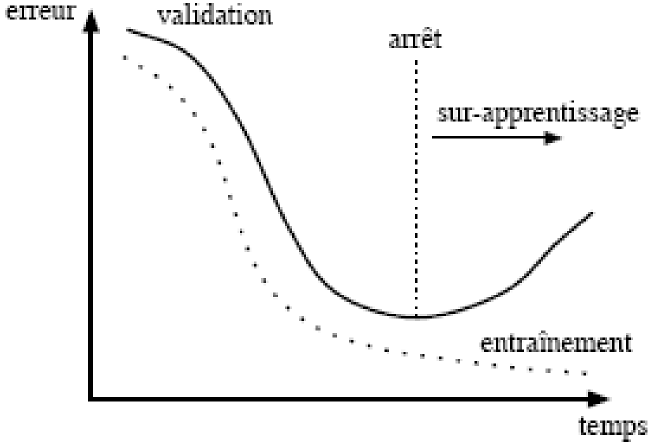
suggestion de décision

Neural networks

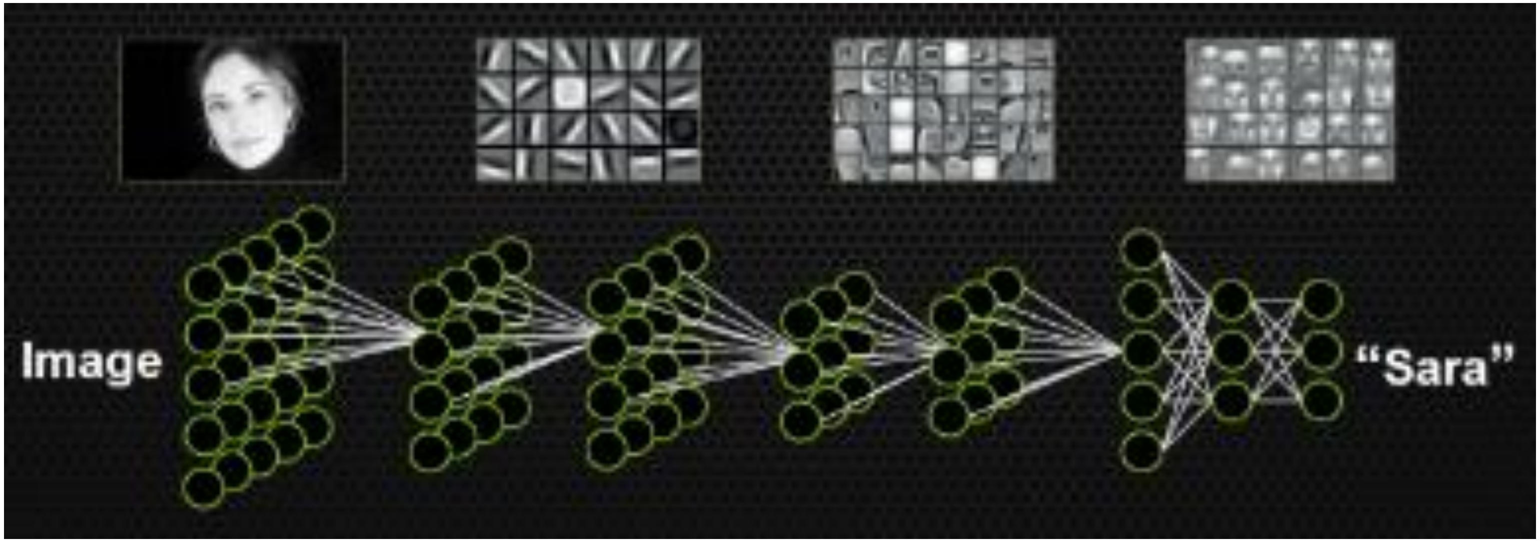


Tricks

- Favour simple NN (you can add the structure in the error)
  - Few layers are enough (theoretically only one)
- Exploit cross validation…



The revenge of Neural Networks: Deep Learning



Automatic description of images

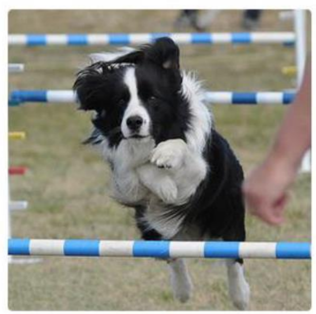

"man in black shirt is playing guitar."


"construction worker in orange safety vest is working on road."


"two young girls are playing with lego toy."


"boy is doing backflip on wakeboard."


"girl in pink dress is jumping in air."


"black and white dog jumps over bar."


"young girl in pink shirt is swinging on swing."


"man in blue wetsuit is surfing on wave."

Andrej Karpathy and Li Fei-Fei, http://cs.stanford.edu/people/karpathy/deepimagesent/

Beyond the Multi-Layer Perceptron

You can learn any function with one hidden layer,
but it's not the best way to do it

Convolution layer for images:

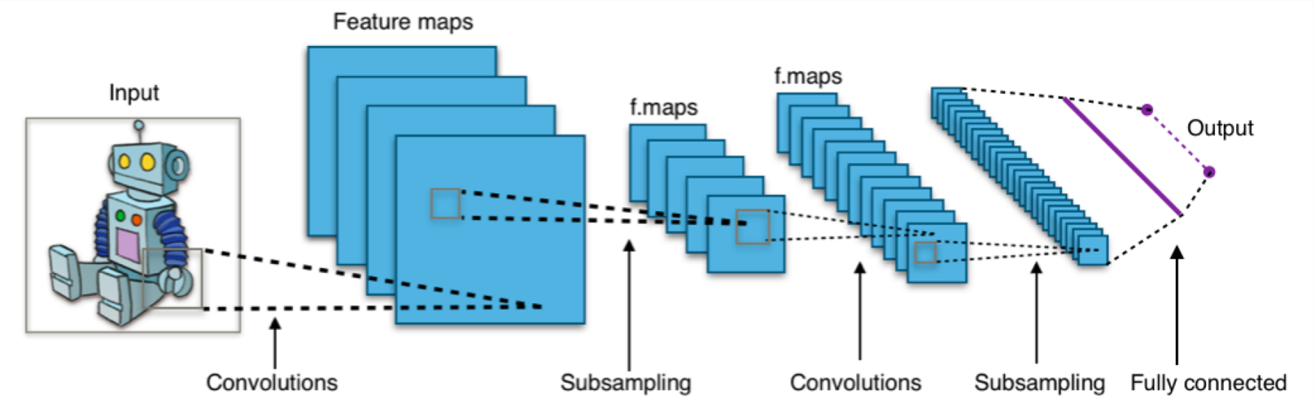


Image by Aphex34 - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=45679374

5 weird tricks to improve training

- How to initialize the model
- How to choose a nonlinearity
- How to avoid over-fitting
- How to pre-process the data
- …

Easy to use !

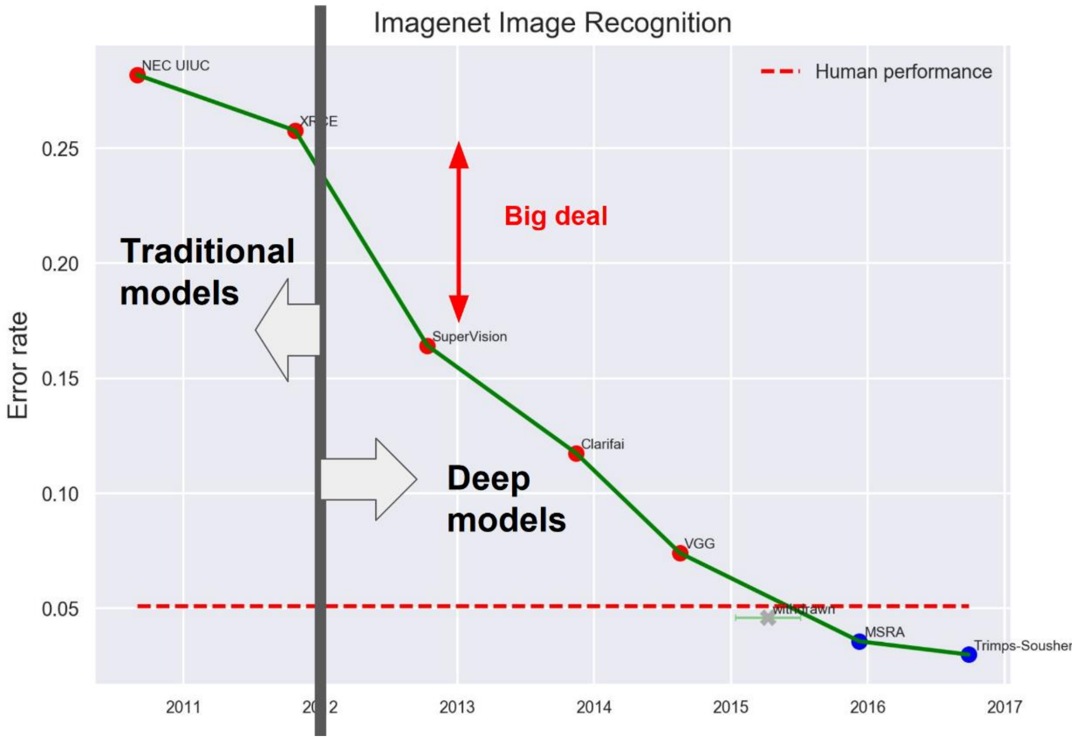
All tips included

TensorFlow

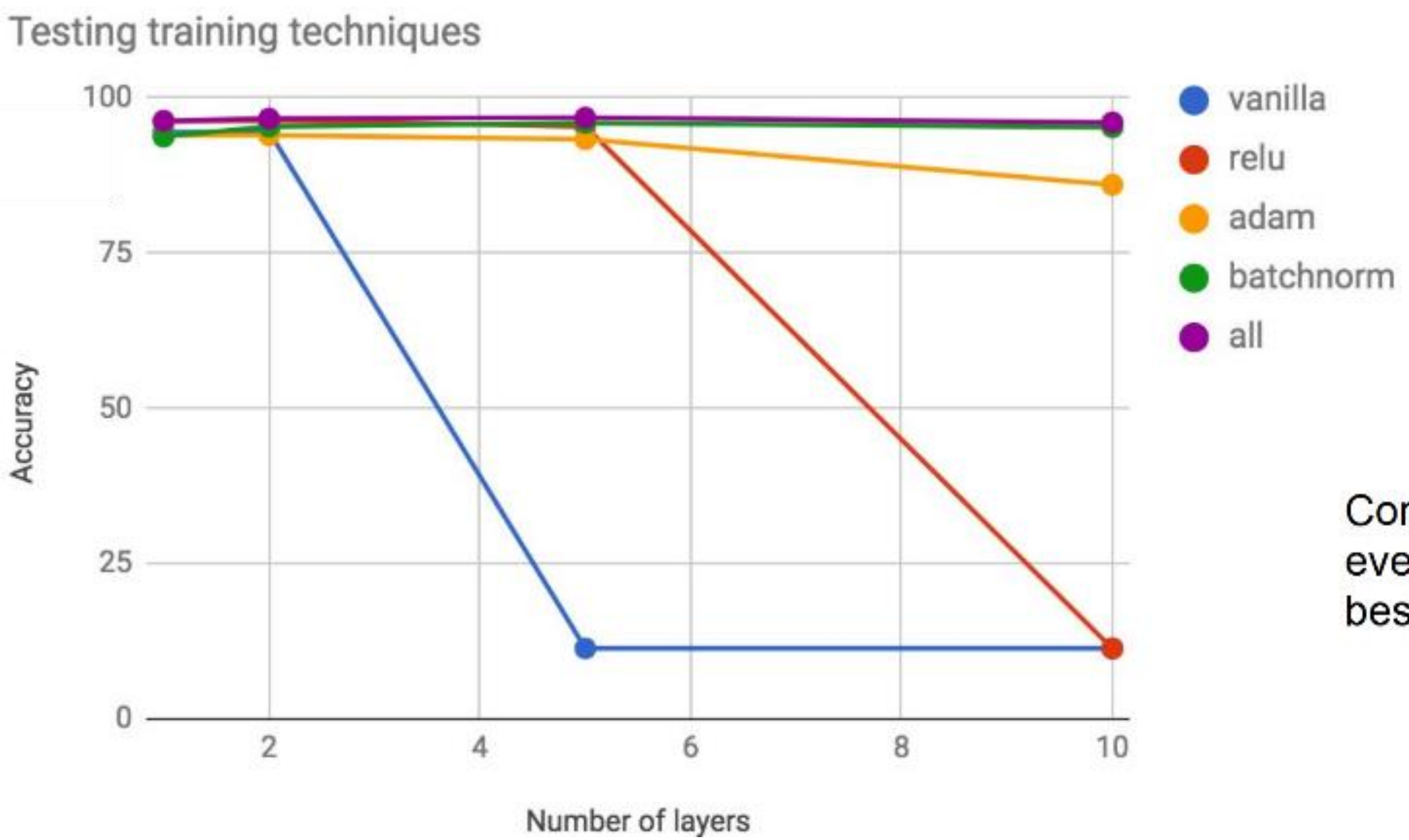
Compute gradients for you

Works on GPU

Theano is a good alternative to TensorFlow



Breakthrough



Combining everything gives best results

Deep in time

Beyond the Multi-Layer Perceptron

Gated memory for sequences:

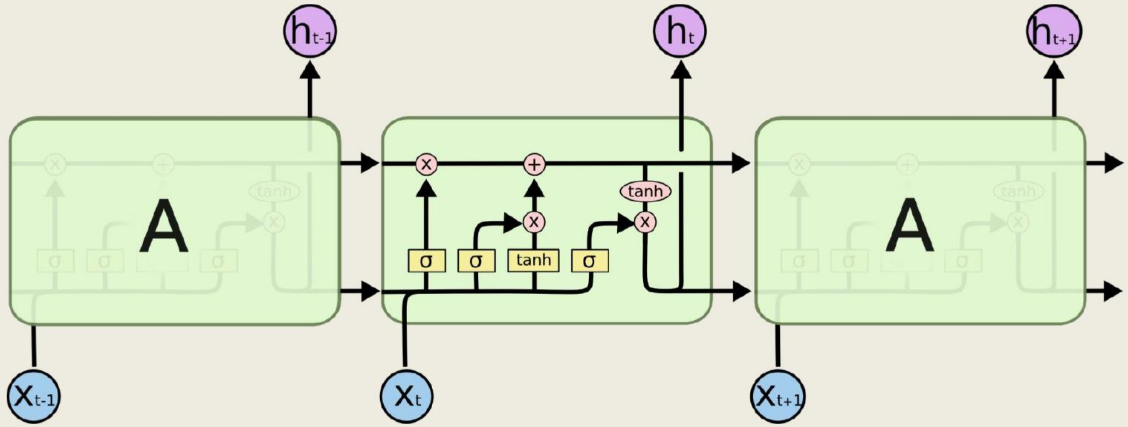
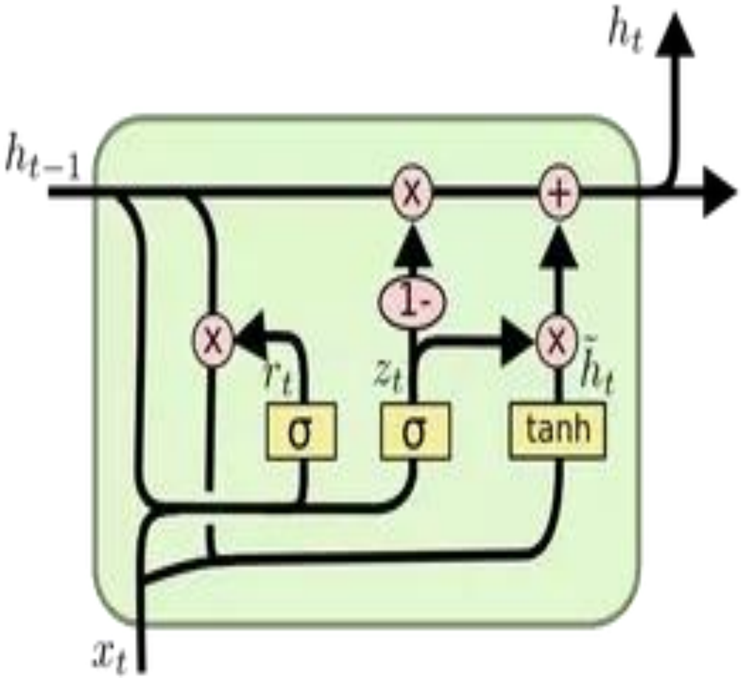


Image by Chris Olah, http://colah.github.io/posts/2015-08-Understanding-LSTMs/
(great explanation of modern recurrent neural nets)

The Neural Network Zoo: http://www.asimovinstitute.org/neural-network-zoo/

GRU RNN



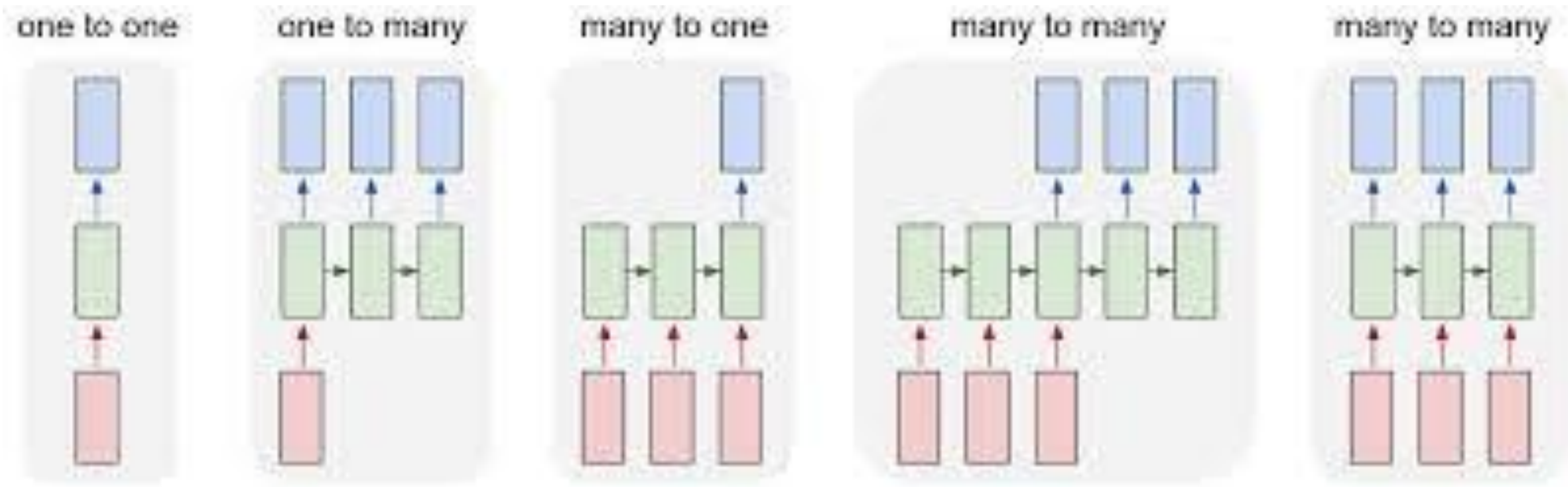
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

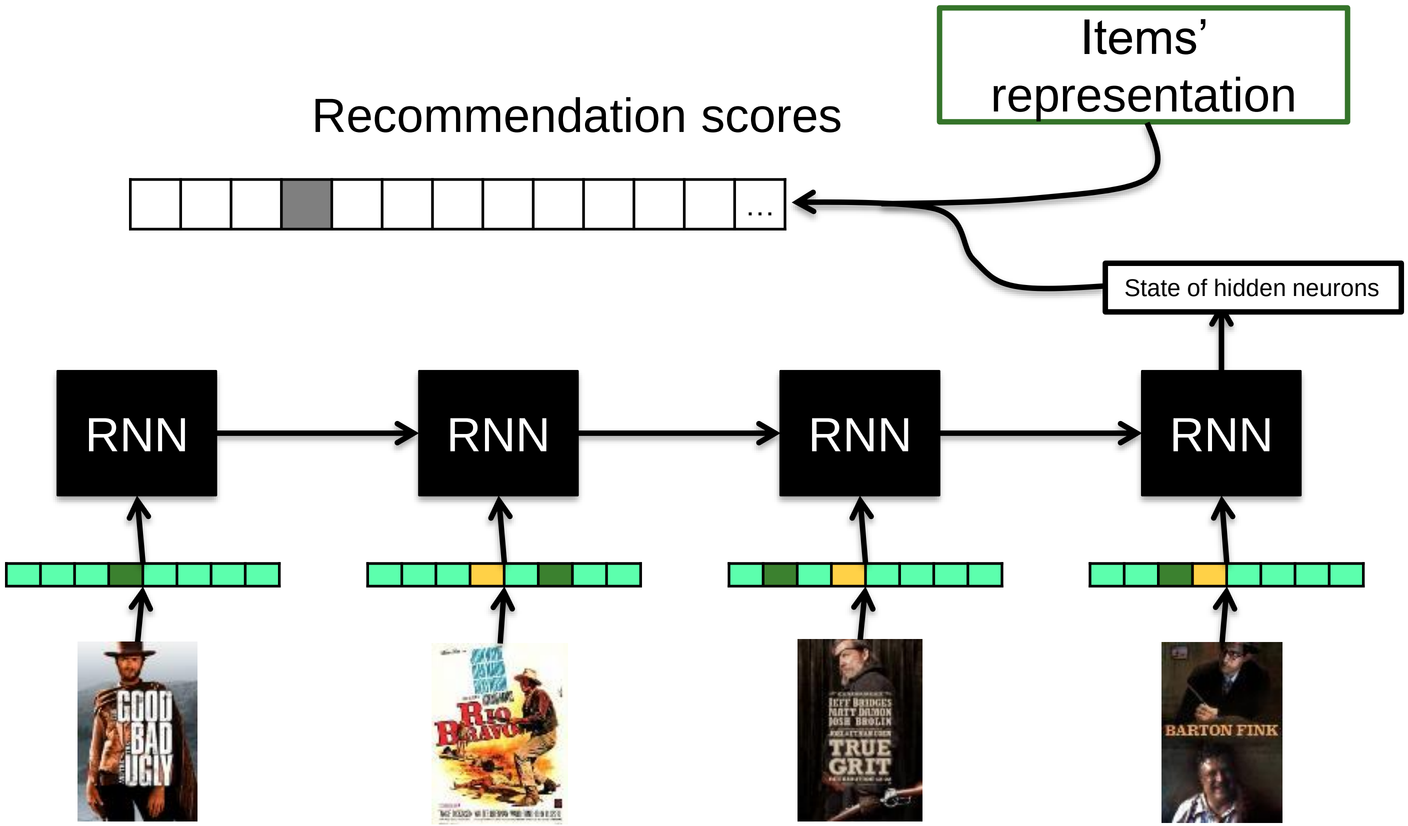
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Different usages



RNN: Algorithmes prédictifs



		Metrics (@10)				
	Method	sps (%)	Item coverage	User coverage (%)	rec (%)	Blockbusters share (%)
Movielens	POP	5.0	50	65.6	3.62	97.73
	BPR-MF	12.44 ± 1.26	388.50 ± 14.32	82.94 ± 1.27	5.58 ± 0.12	44.08 ± 1.43
	User KNN	14.40	277	80.8	6.31	50.36
	MC	29.20	518	77.0	4.90	16.47
	FPMC	28.10 ± 0.71	614.00 ± 10.32	82.94 ± 1.45	5.70 ± 0.13	14.26 ± 0.71
	RNN-CCE	33.45 ± 1.17	669.75 ± 15.58	87.73 ± 0.98	7.52 ± 0.14	13.91 ± 0.28
	RNN-Blackout	33.68 ± 1.51	657.75 ± 40.38	87.47 ± 1.32	7.45 ± 0.12	13.42 ± 1.29
	RNN-BPR	26.76 ± 4.04	610.33 ± 80.56	87.29 ± 2.60	7.26 ± 0.66	16.50 ± 3.23
	RNN-TOP1	25.73 ± 1.52	386.67 ± 23.39	88.78 ± 0.70	7.75 ± 0.30	27.61 ± 1.78
	RNN-Hinge	30.91 ± 2.15	611.56 ± 45.52	88.40 ± 0.82	7.72 ± 0.17	16.40 ± 1.92
Netflix	POP	5.92	54	68.2	4.65	100
	BPR-MF	9.93 ± 0.70	591.60 ± 16.36	79.19 ± 0.71	6.88 ± 0.09	73.84 ± 1.32
	User KNN	13.04	383	80.84	8.49	79.86
	MC	32.50	594	64.50	3.17	53.02
	FPMC	26.38 ± 0.80	542.56 ± 6.87	70.98 ± 0.46	3.75 ± 0.08	64.53 ± 0.51
	RNN-CCE	41.00 ± 0.99	845.88 ± 26.00	82.43 ± 0.48	6.37 ± 0.16	53.15 ± 1.39
	RNN-Blackout	35.36 ± 1.26	778.25 ± 53.22	77.26 ± 1.16	4.84 ± 0.21	50.49 ± 2.91
	RNN-BPR	32.20 ± 0.30	791.67 ± 10.66	83.02 ± 0.74	6.00 ± 0.08	55.90 ± 0.87
	RNN-TOP1	23.77 ± 0.33	436.50 ± 0.50	78.79 ± 0.51	5.28 ± 0.08	75.19 ± 0.57
	RNN-Hinge	32.28 ± 0.87	935.20 ± 38.23	78.92 ± 0.65	5.76 ± 0.24	39.29 ± 0.43
RSC15	POP	1.24	11	2.24	1.2	100
	MC	39.50	2945	46.55	32.76	33.33
	RNN-CCE	52.78 ± 0.08	3536.67 ± 4.99	59.34 ± 0.10	44.52 ± 0.06	33.14 ± 0.17
	RNN-Blackout	40.08 ± 1.24	2721.33 ± 88.19	46.83 ± 1.22	33.36 ± 1.11	34.68 ± 0.35
	RNN-BPR	33.97 ± 0.34	2105.67 ± 9.46	41.73 ± 0.49	28.88 ± 0.52	40.91 ± 0.48
	RNN-TOP1	17.57 ± 0.43	673.67 ± 13.22	23.50 ± 0.71	15.37 ± 0.45	62.60 ± 0.31
	RNN-Hinge	35.12 ± 0.27	2374.33 ± 21.93	43.18 ± 0.16	30.24 ± 0.23	33.76 ± 0.30

In Proceedings of NIPS 2016 and UMAP 2017