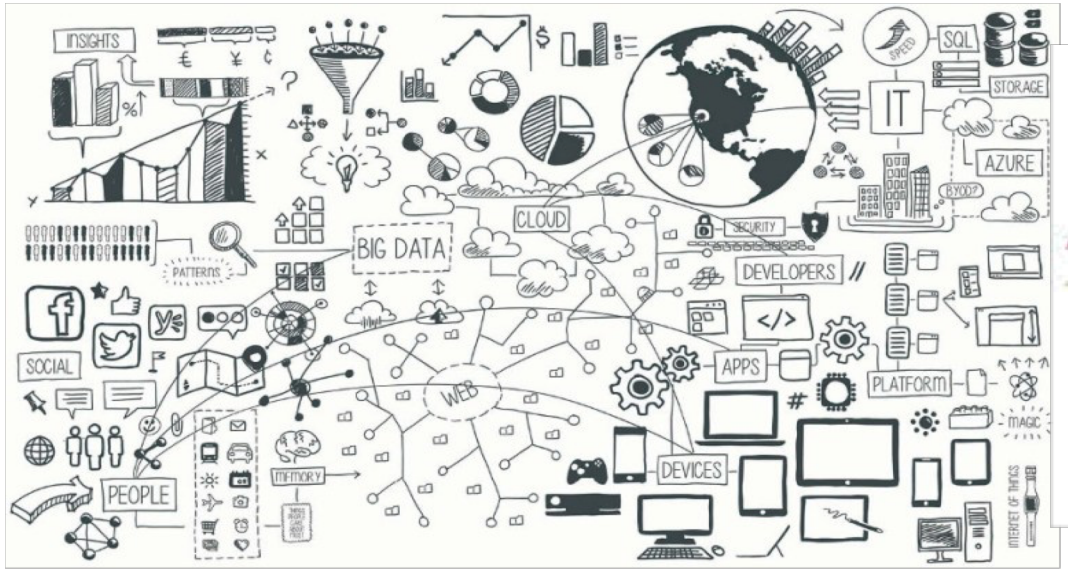
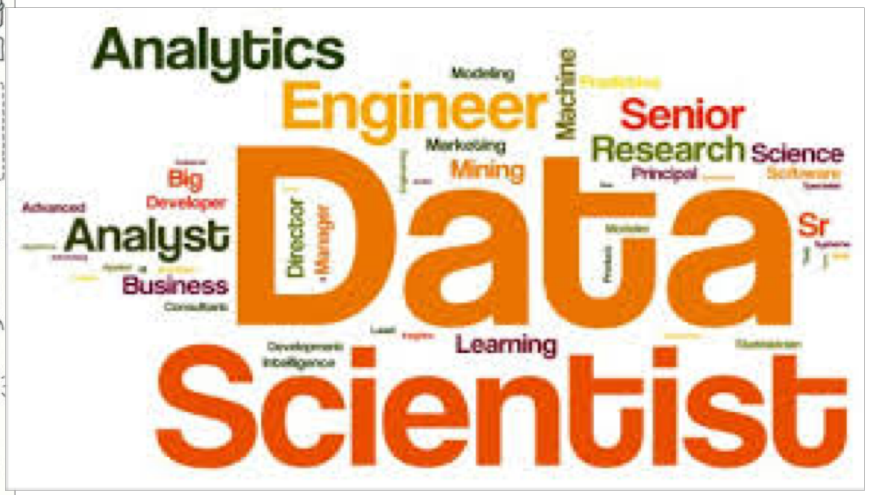
Machine Learning I (Neural Networks)

Reservoir Computing and Extreme Learning Machines



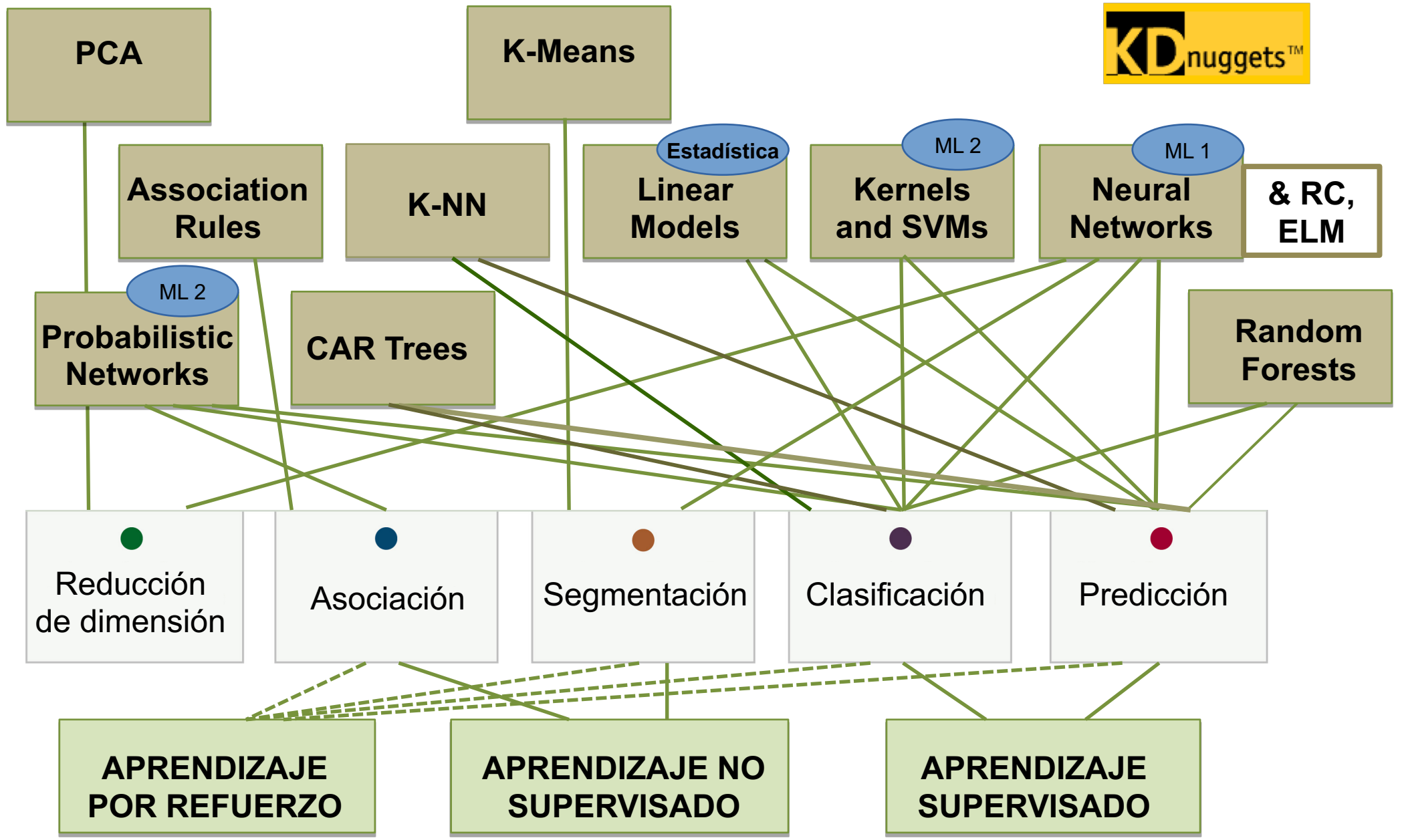


José Manuel Gutiérrez
Jorge Baño

Grupo de Meteorología

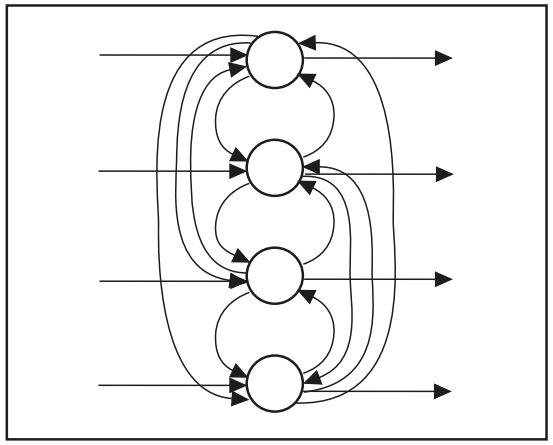
Univ. de Cantabria – CSIC
MACC / IFCA

https://www.kdnuggets.com/2017/10/top-10-machine-learning-algorithms-beginners.html



Supervised Problems. **Input-Input pairs are provided:**
(x1,x1), (x2,x2), ..., (xn,xn) and the network learns $x = f(x+\varepsilon)$.

Autoassociative memories (Hopfield).
Single layer with lateral delayed connections.



Pattern Recognition
OCR, images
Memories (robust to noise)

Prediction: Input => Input

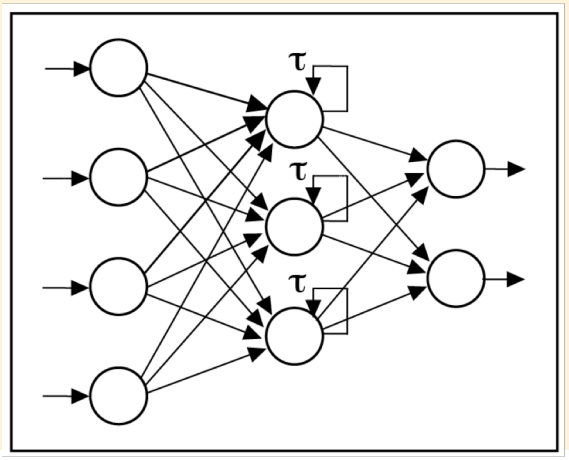
Learning: Hegg

Autoencoders (later)

Feature extraction, compression.

Supervised Problems (with memory). **Input-Output pairs are provided:**
(x1,y1), (x2,y2), ..., (xn,yn) and the network learns $y_t = f(x_{t-1,t-2,\ldots}+\varepsilon)$.

Recurrent Networks or Elman/Jordan nets.
Multilayer network with hidden/output delayed lines.

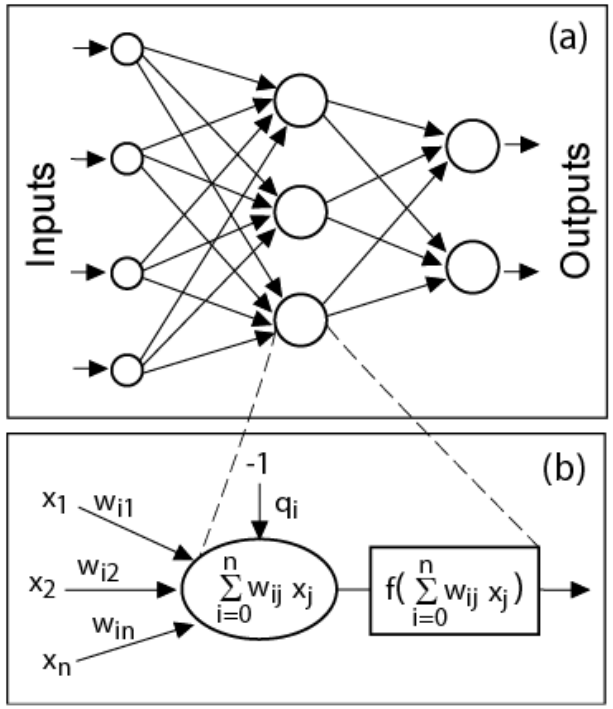


Time series analysis
Video, natural language
Interpolation and fitting

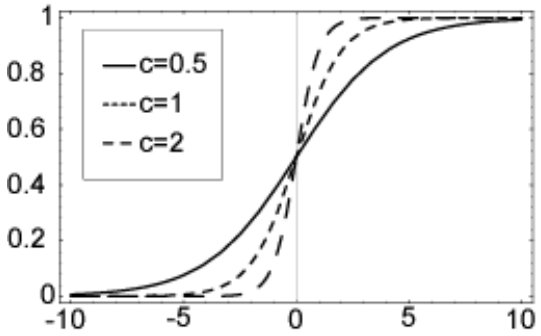
Prediction: Input => Output

Learning: Backpropagation in time

x : h : y



The neural activity (output) is given by a nonlinear function.



$$y_i = f(\sum_k W_{ik} f(\sum_j w_{kj} a_{pj}))$$ **h_i**

$$E(w) = \frac{1}{2}\sum_{p,i}(b_{pi} - f(\sum_k W_{ik}\, f(\sum_j w_{kj} a_{pj})))^2$$

Gradient descent

$$\Delta W_{ik} = -\eta \frac{\partial E}{\partial W_{ik}};\ \Delta w_{kj} = -\eta \frac{\partial E}{\partial w_{kj}},$$

1. Init the neural weight with random values
2. Select the input and propagate it (estimate hidden and output)
3. Compute the error associate with the output

$$\delta_{pi} = (b_{pi} - \hat{b}_{pi}) f'(\hat{B}_{pi})$$
$$= (b_{pi} - \hat{b}_{pi})\hat{b}_{pi}(1 - \hat{b}_{pi})$$

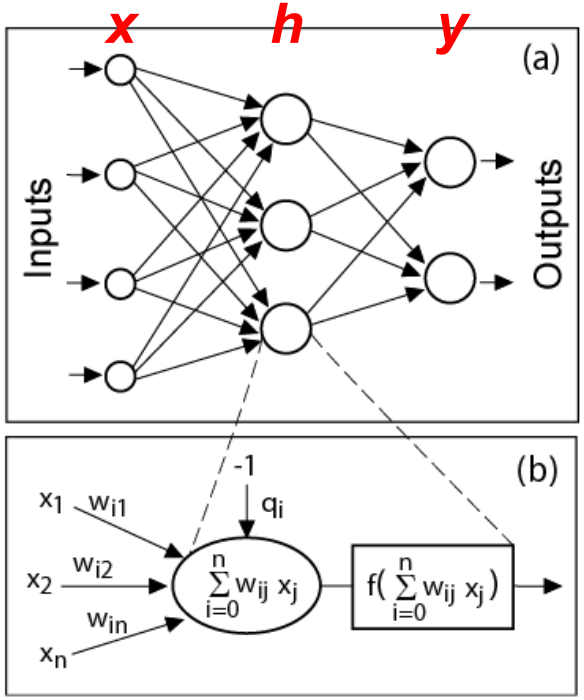
4. Compute the error associate with the hidden neurons

$$\psi_{pk} = \sum_i \delta_{pi}\, W_{ki}\, f'(\hat{H}_{pk})$$

5. Compute

$$\Delta W_{ik} = \eta\ \delta_{\mathrm{pi}}\, \hat{h}_{pk},\ \ \Delta w_{kj} = \eta\, \psi_{pk}\, a_{pj}$$

and update the neural weight according to these values



$$y_j = f\Big(\sum_i \beta_{ji}\, f\Big(\sum_k \alpha_{ik}\, x_{kp}\Big)\Big)$$

h_i

The input-to-hidden weights are randomly initilialized. The optmization problem is a linear one.

Algorithm ELM : Given a training set $\aleph = \{(\mathbf{x}_i, \mathbf{t}_i) | \mathbf{x}_i \in \mathbf{R}^n, \mathbf{t}_i \in \mathbf{R}^m, i = 1, \cdots, N\}$, activation function $g(x)$, and hidden neuron number $\tilde{N}$,

step 1: Assign arbitrary input weight $\mathbf{w}_i$ and bias b_i, $i = 1, \cdots, \tilde{N}$.

step 2: Calculate the hidden layer output matrix $\mathbf{H}$.

step 3: Calculate the output weight β:

$$\mathbf{H}\beta = \mathbf{T} \qquad \beta = \mathbf{H}^{\dagger}\mathbf{T} \qquad (16)$$

Int. J. Mach. Learn. & Cyber. (2011) 2:107–122
DOI 10.1007/s13042-011-0019-y

ORIGINAL ARTICLE

Extreme learning machines: a survey

Guang-Bin Huang · Dian Hui Wang · Yuan Lan

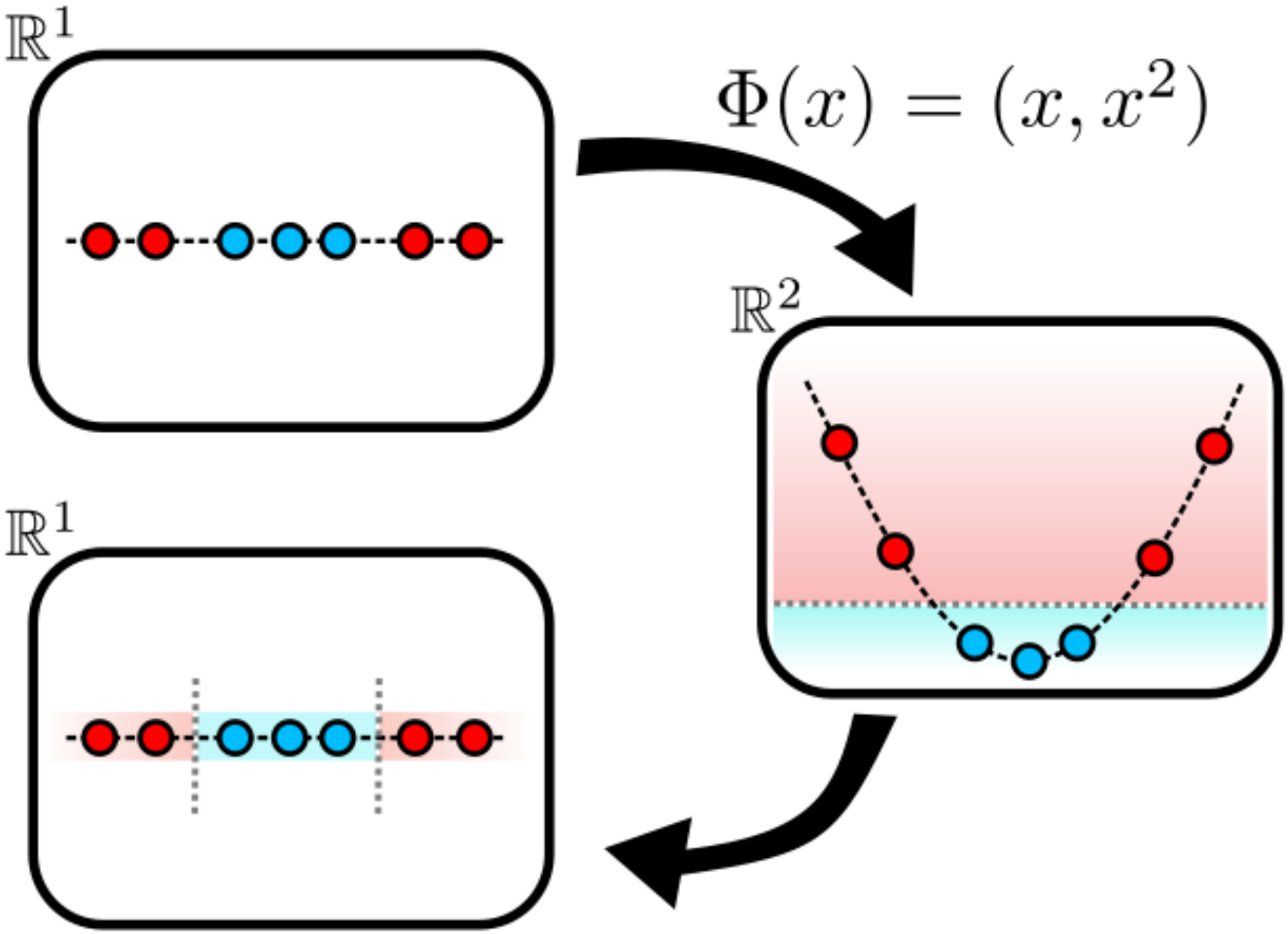
Definition 2.1: [10] A matrix $\mathbf{G}$ of order $n \times m$ is the Moore-Penrose generalized inverse of matrix $\mathbf{A}$ of order $m \times n$, if

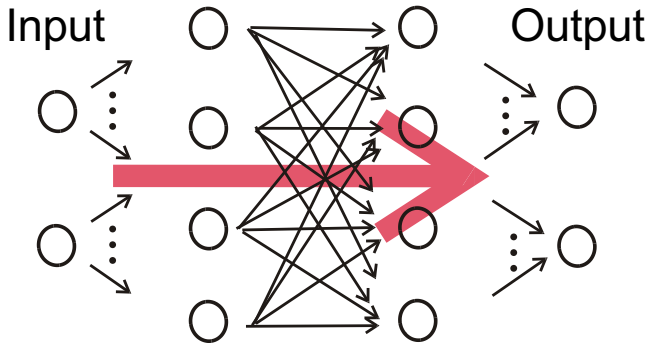
$$\mathbf{AGA} = \mathbf{A}, \mathbf{GAG} = \mathbf{G}, (\mathbf{AG})^T = \mathbf{AG}, (\mathbf{GA})^T = \mathbf{GA}$$

➔ Provides LS solution with minimum norm.

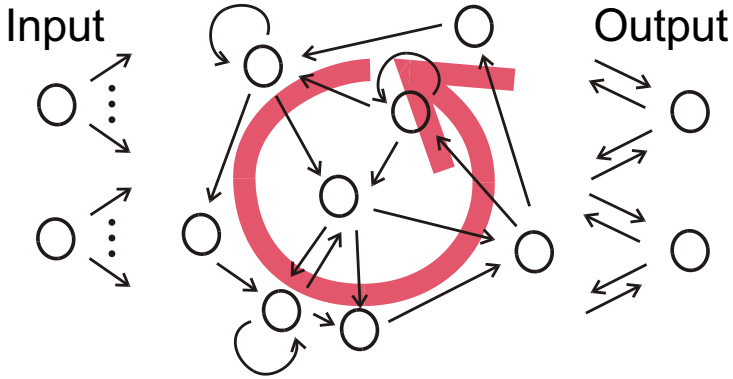
$$\sum_{j=1}^{\tilde{N}} \|\mathbf{o}_j - \mathbf{t}_j\|$$

The problem is now linearly separable in a nonlinear subspace of the original one.

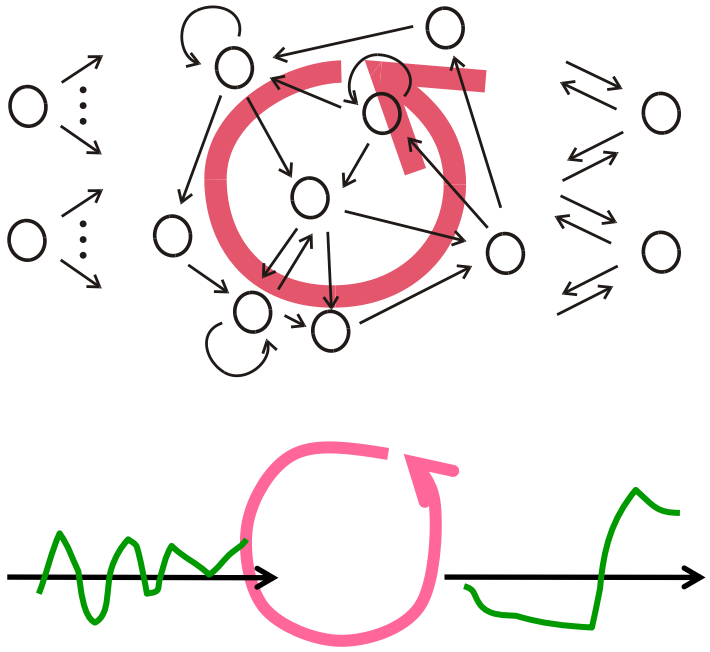




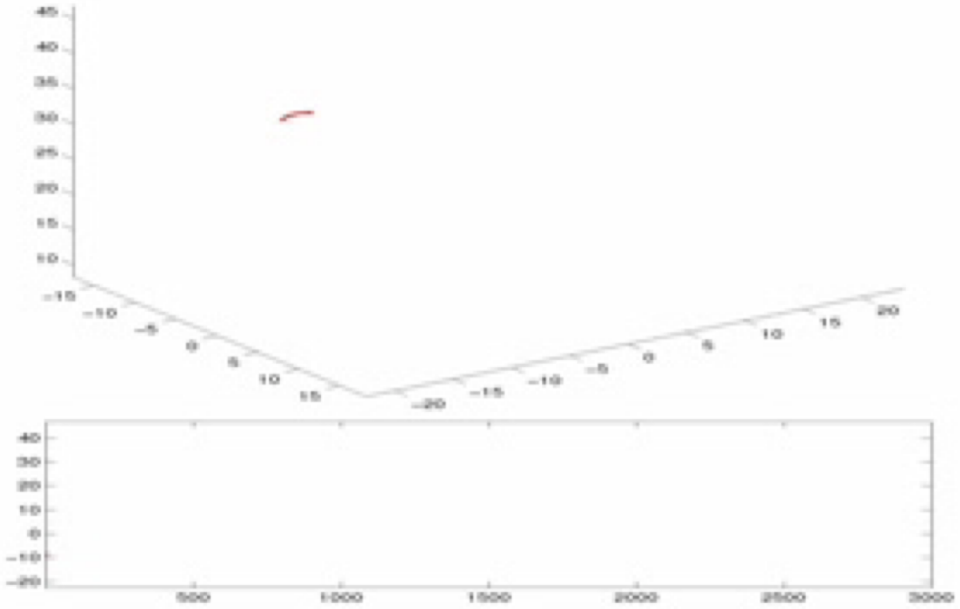
- connections only "from left to right", no connection cycle
- activation is fed forward from input to output through "hidden layers"
- no memory

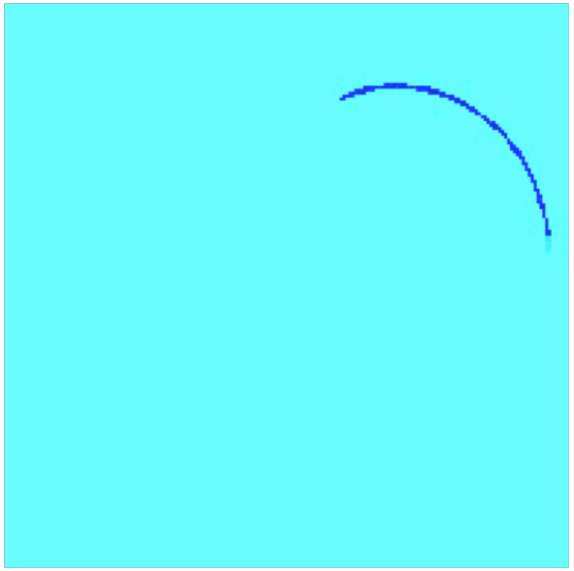


- at least one connection cycle
- activation can "reverberate", persist even with no input
- system with memory



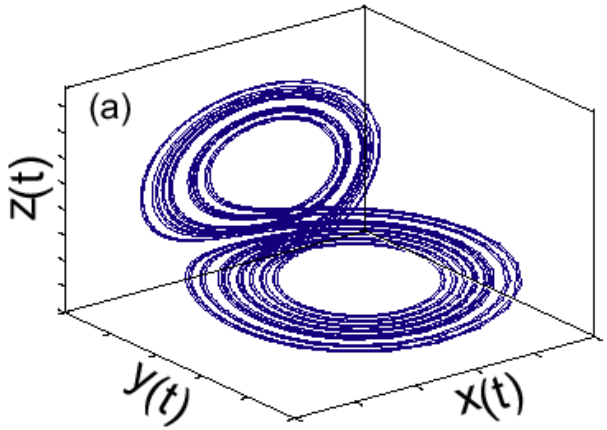
- input time series → output time series
- can approximate any dynamical system (universal approximation property)
- mathematical analysis difficult
- learning algorithms computationally expensive and difficult to master
- few application-oriented publications, little research

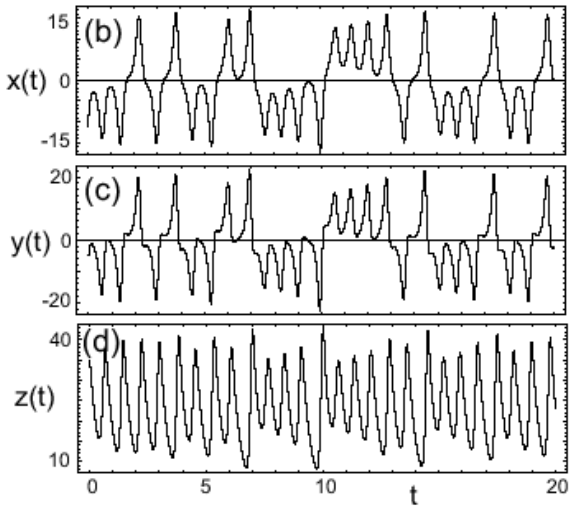




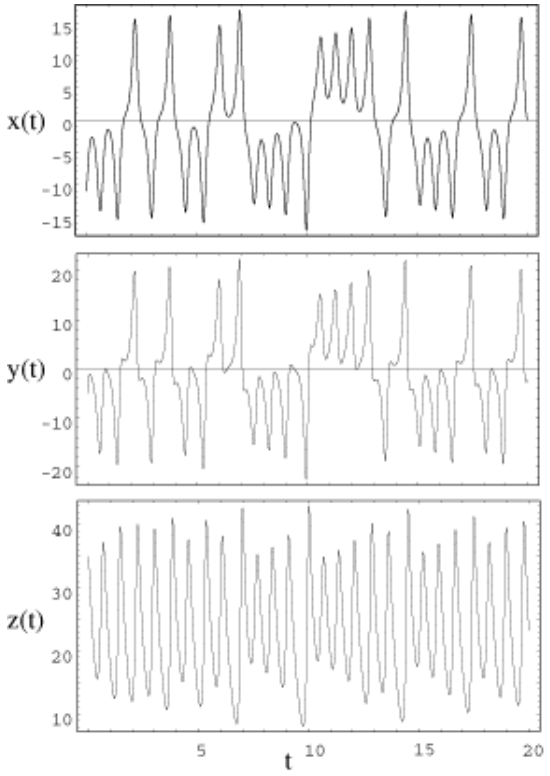
Los modelos atmosféricos son “no lineales” y, por tanto, sensibles a pequeñas perturbaciones en las condiciones iniciales (Caos, Lorenz 1963).

$$\begin{cases} \dfrac{dx}{dt} = a(y-x) \\ \dfrac{dy}{dt} = x(b-z)-y \\ \dfrac{dz}{dt} = xy - cz \end{cases}$$





Dada una serie temporal de 2000 puntos (t=20), se prueban distintos modelos de red para comprobar el poder modelizador.



Tres variables
(x,y,z)

(x_n, y_n, z_n)

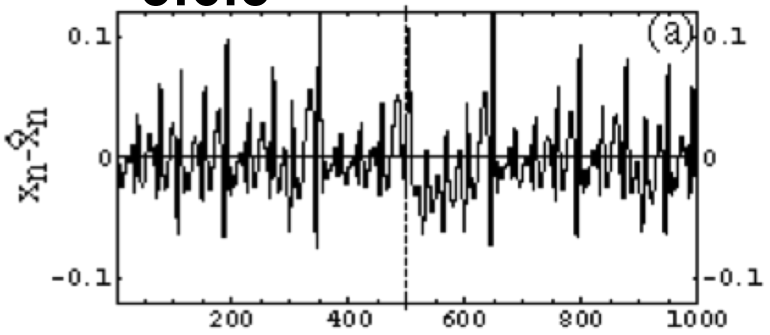
$\downarrow$

$(x_{n+1}, y_{n+1}, z_{n+1})$

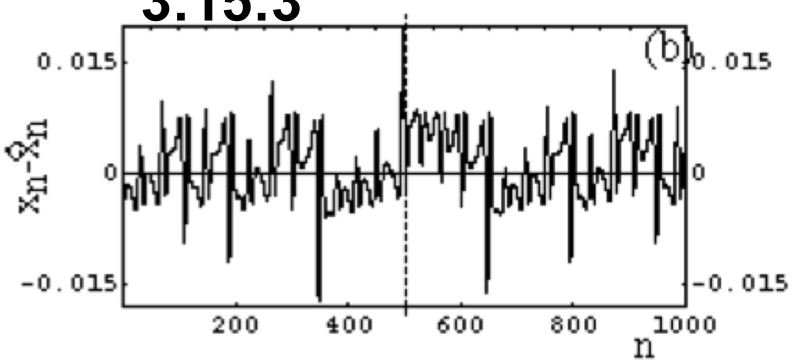
Sistema contínuo
Red neuronal

3:k:3

3:6:3



3:15:3

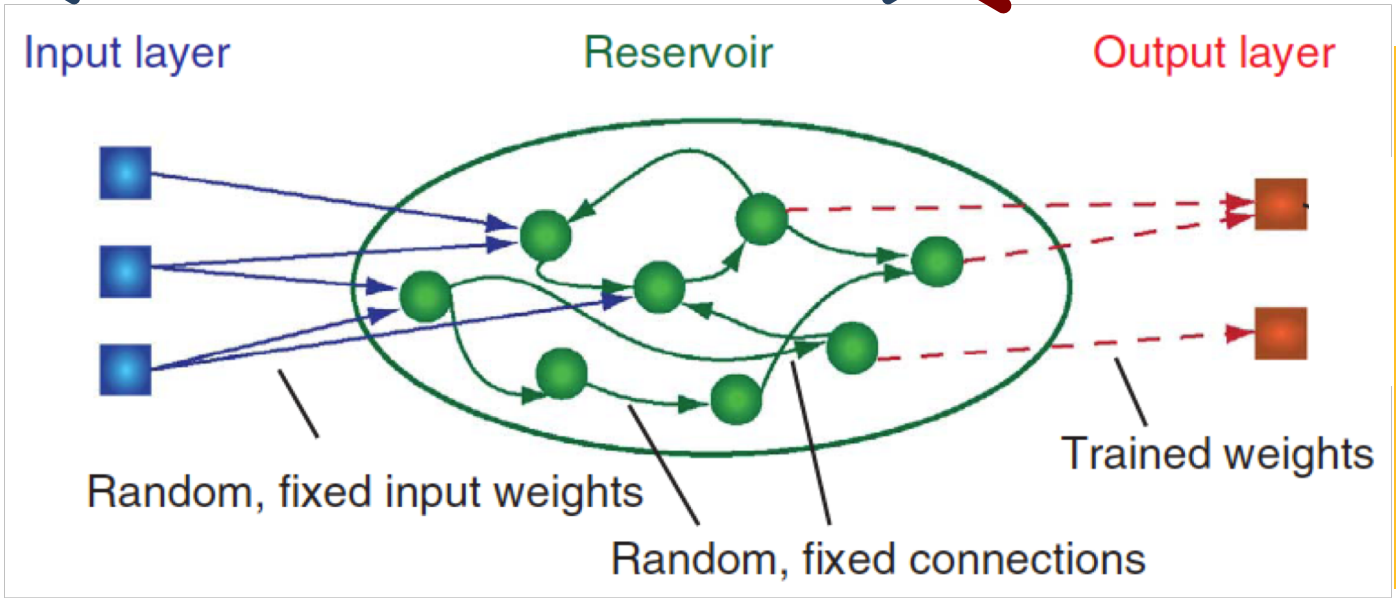


$$\hat{x}_{n+1} = -3768.18 - \frac{0.34}{1+e^{9.31+0.53\,x_n-0.68\,y_n-0.21\,z_n}} + \frac{0.92}{1+e^{7.64-0.121\,x_n-0.149\,y_n-0.13\,z_n}} - \frac{2.75}{1+e^{6.19+0.15\,x_n+0.0451\,y_n-0.09\,z_n}} - \frac{2.04}{1+e^{1.13+0.06\,x_n+0.0119\,y_n-0.06\,z_n}} + \frac{7164.31}{1+e^{-0.12+0.00021\,x_n-0.0002\,y_n+0.000021\,z_n}} - \frac{63.52}{1+e^{-0.24+0.08\,x_n-0.016\,y_n+0.0049\,z_n}},$$

RC models (e.g. Echo State Networks, ESN) are supervised (input-output) machine learning tools which are built in two steps:

Building the reservoir (high-dimensional recurrent nonlinear network) and mapping the input to the reservoir

Fitting the output to the reservoir using a simple linear model.



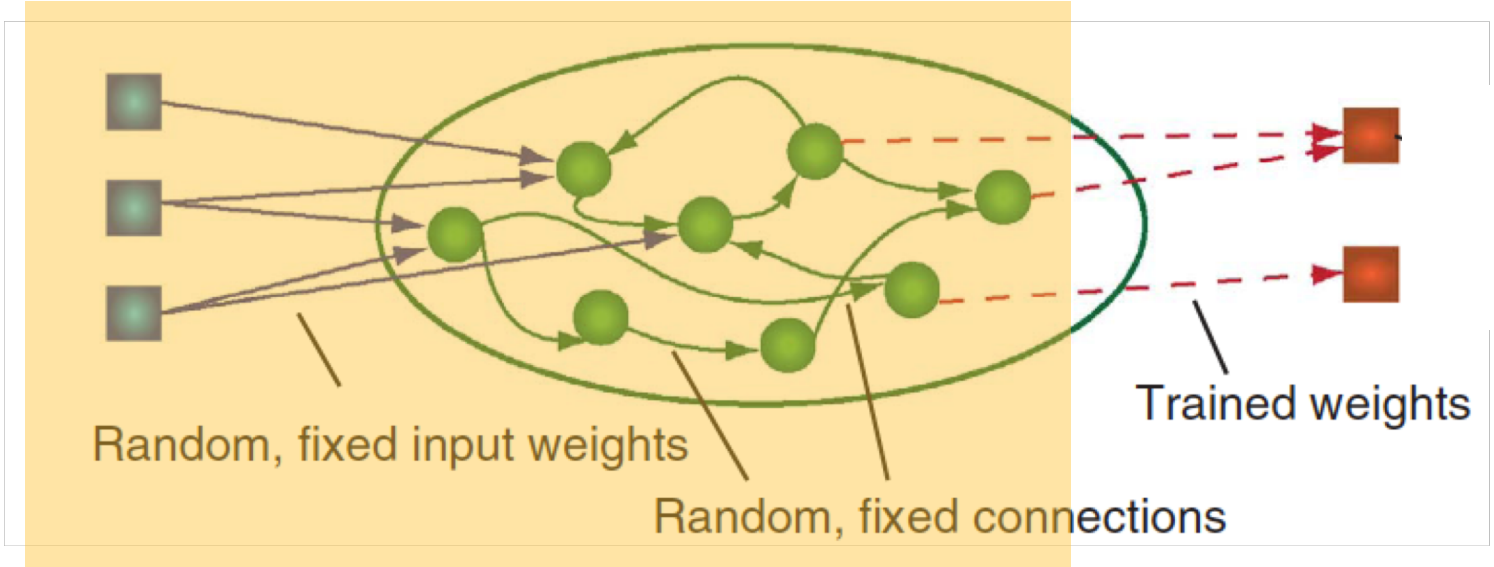
¡Only the output connections are trained !!

So the final optimization can be easily solved

RC has been successfully applied in many tasks, including time series prediction, speech recognition, and pattern classification. These tasks require **memory** and **nonlinearity**.

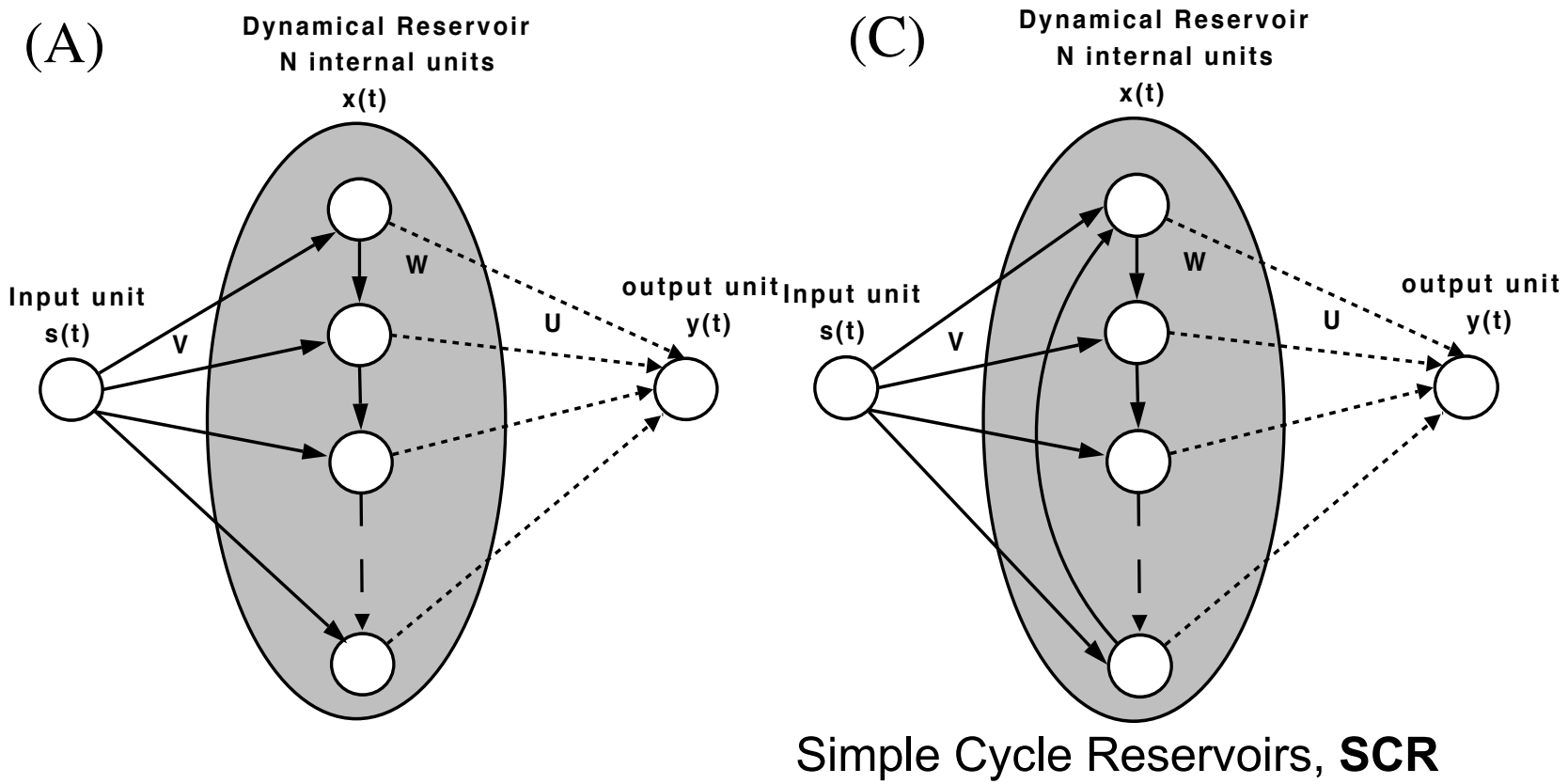
However, there are several problems preventing RC to become widely adopted for machine learning problems.

- **The input- and reservoir-weights (including network connectivity) are randomly chosen**. Thus, **performance relies on <u>trial and error</u>** (testing different model realizations).



- **Some properties of the reservoir are poorly understood**; i.e. It is used as a **<u>black-box technique</u>**.

It has been recently shown that simple deterministically constructed reservoirs solve some of these problems with **similar performance than standard ESNs** (in some tested tasks).

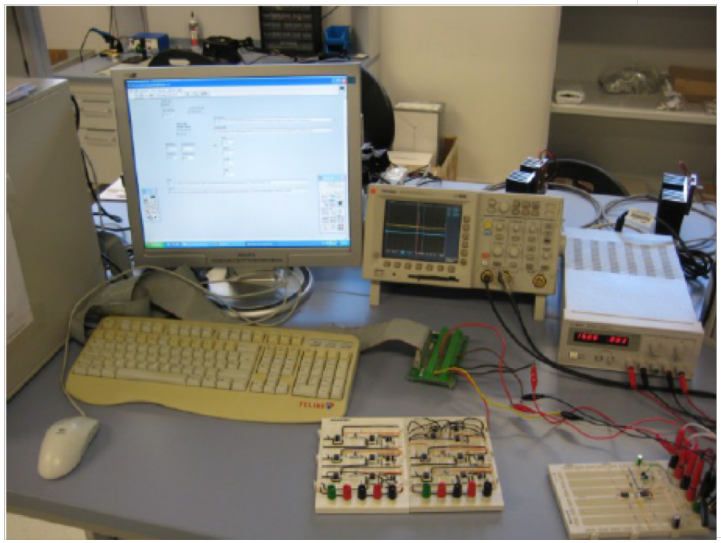


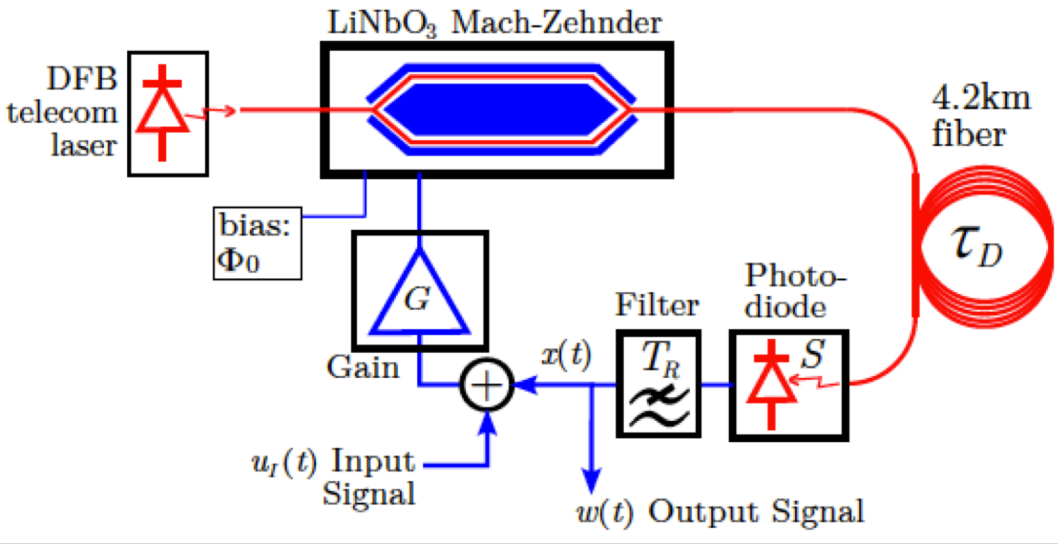
Simple Cycle Reservoirs, **SCR**

A. Rodan and P. Tino, **Minimum Complexity Echo State Network**, IEEE Transactions on Neural Networks, 22, 131-144, 2011.

phocus

towards a PHOtonic liquid state machine based on delay-CoUpled Systems



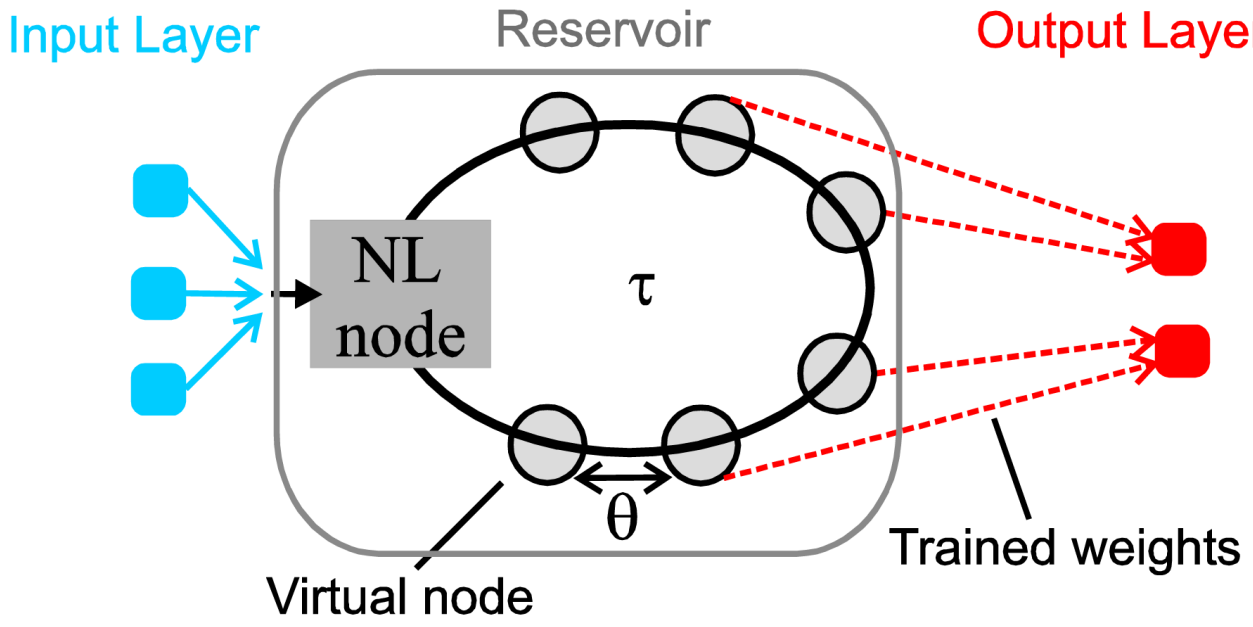


Appeltant et al. (2011), **Information processing using a single dynamical node as complex system**, *Nature Communications*, DOI 10.1038/ncomms1476.

Larger et al. (2012), **Photonic information processing beyond Turing: An optoelectronic implementation of reservoir computing**, *Optics Express*, 20(3), 3241.

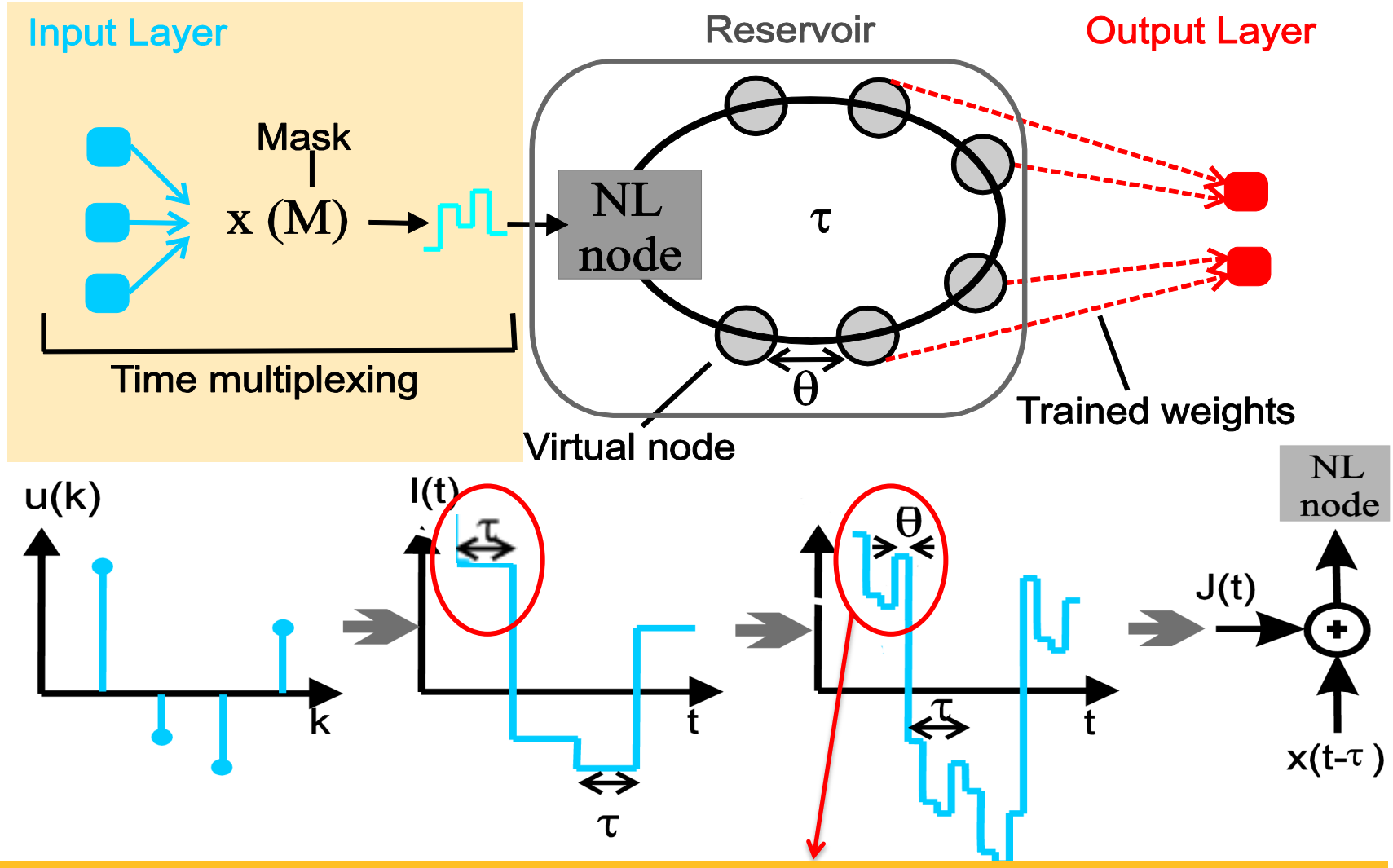
A high-dimensional reservoir of virtual nodes is achieved by:

1. Dividing the delay interval into N pieces of length $\theta=\tau/N$.
1. Their endpoints represent virtual nodes: They are simply a delayed version of the dynamical variable x(t).



- The **reservoir weights** are derived from the physical counterpart (they are determined by the feedback ρ and the input γ strengths).
- **The readout function** is obtained as in standard reservoirs.

The addition of an external input induces a complex transient response, allowing for efficient computation.



The mask "provides" random input connections to the virtual nodes.