A Brief Introduction to Artificial Neural Networks

Dr. José Ernesto Rayas Sánchez

1

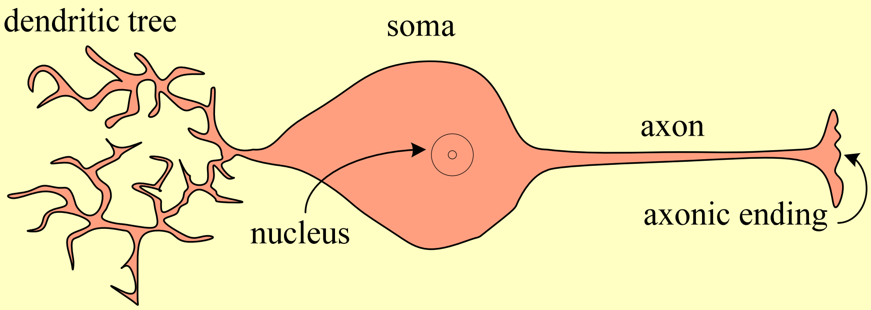
Outline

- Biological neurons
- Synapses
- Definitions of artificial neural networks
- Types of artificial neurons
- Types of activation functions
- Types of ANN paradigms
- 3-Layer perceptrons
- Future directions and conclusions

Dr. J. E. Rayas Sánchez

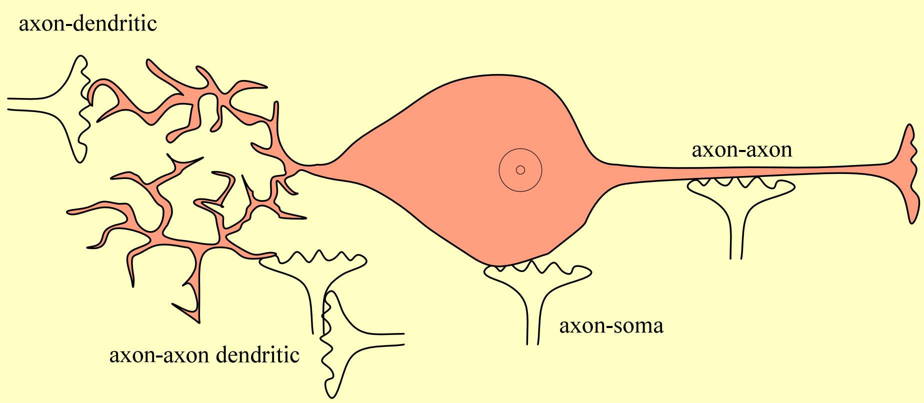
2

Biological Neuron



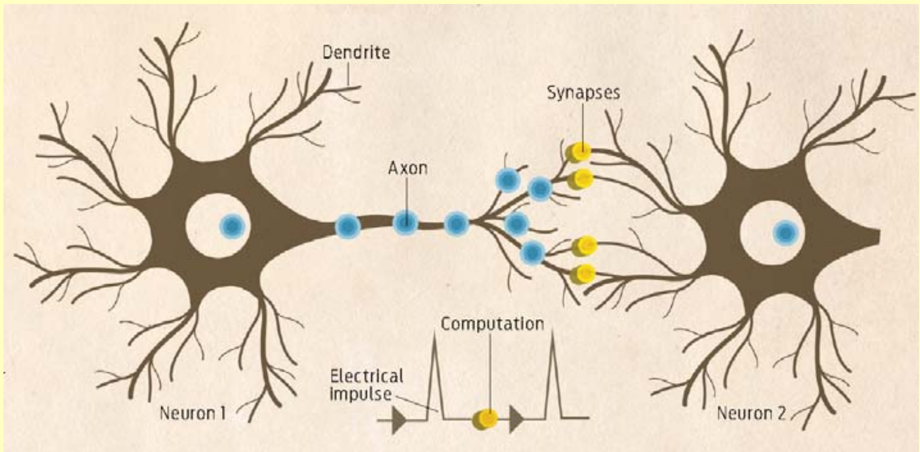
Dr. J. E. Rayas Sánchez *(Kartalopoulos, 1996)* 3

Types of Synapses



Dr. J. E. Rayas Sánchez *(Kartalopoulos, 1996)* 4

Biological Neurons – Computation and Storage



(Versace et al., IEEE Spectrum 2010)

Artificial Neural Networks (ANNs)

- An Artificial Neural Network (ANN) is a massively parallel distributed processor made up of simple processing units (neurons), that is able of acquiring knowledge from its environment though a learning process
- ANNs are also information processing systems that emulate biological neural networks: they are inspired in the ability of human brain to learn from observation and generalize by abstraction

Artificial Neurons

- An artificial neuron is a simple processing unit that receives and combines signals from many other neurons
- Common types of artificial neurons are:

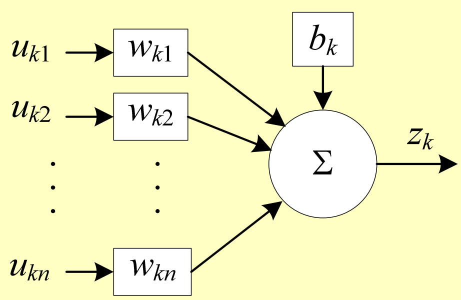
  Linear Neurons

  Inner-Product Nonlinear Neuron

  Euclidean Distance Neuron

Dr. J. E. Rayas Sánchez 7

Linear Neuron



$$z_k = b_k + \boldsymbol{u}_k^T \boldsymbol{w}_k$$

$\boldsymbol{u}_k = [u_{k1} \dots u_{kn}]^T$ vector of inputs

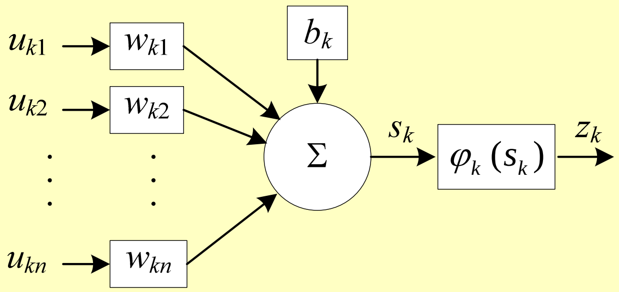
$\boldsymbol{w}_k = [w_{k1} \dots w_{kn}]^T$ vector of weighting factors

b_k bias or offset term

z_k activation potential or induced local field, output signal

Dr. J. E. Rayas Sánchez 8

Inner Product Nonlinear Neuron



$$s_k = b_k + \boldsymbol{u}_k^T \boldsymbol{w}_k$$

$$z_k = \varphi_k(s_k)$$

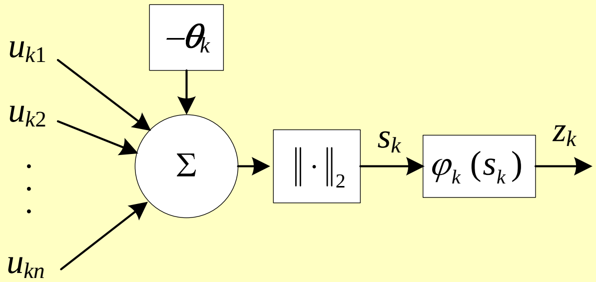
$\boldsymbol{u}_k = [u_{k1} \ldots u_{kn}]^T$ inputs $\quad$ $\boldsymbol{w}_k = [w_{k1} \ldots w_{kn}]^T$ weighting factors

b_k bias or offset term $\quad$ s_k activation potential

$\varphi_k(s_k)$ activation function or squashing function

z_k output signal

Euclidean Distance Neuron



$$s_k = \| \boldsymbol{u}_k - \boldsymbol{\theta}_k \|_2$$

$$z_k = \varphi_k(s_k)$$

$\boldsymbol{u}_k = [u_{k1} \ldots u_{kn}]^T$ inputs

$\boldsymbol{\theta}_k = [\theta_{k1} \ldots \theta_{kn}]^T$ center vector with respect to the Euclidean distance is measured

$\varphi_k(s_k)$ activation function, typically a Gaussian function

z_k output signal

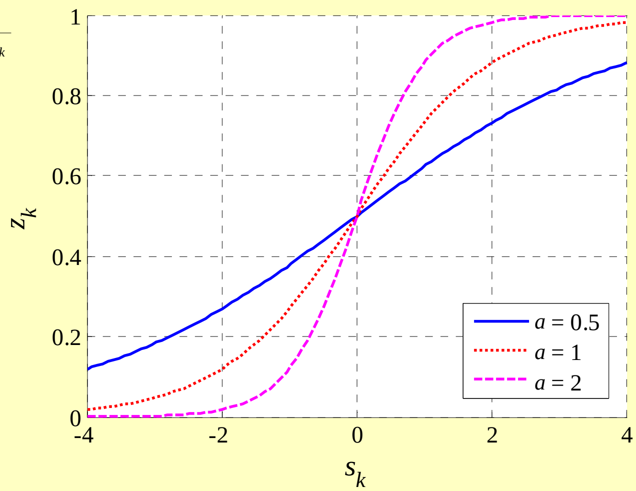
Typical Activation Functions

Sigmoid or logistic function

$$z_k = \varphi_k(s_k) = \frac{1}{1+e^{-s_k}}$$

With a slope parameter

$$z_k = \frac{1}{1+e^{-a\,s_k}}$$



Typical Activation Functions (cont)

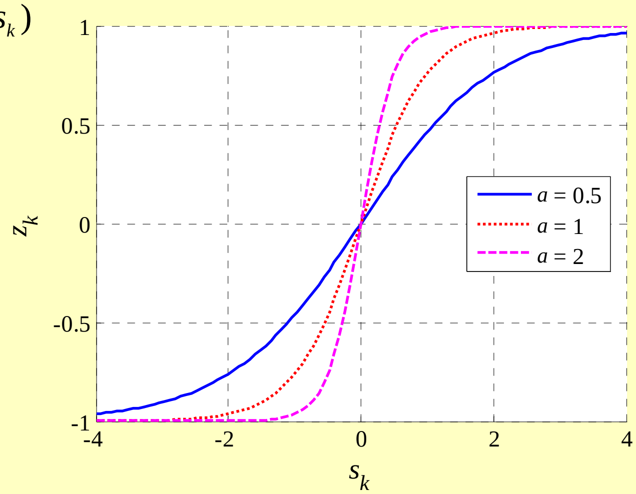
Hyperbolic tangent function

$$z_k = \varphi_k(s_k) = \tanh(s_k)$$

$$\tanh(s_k) = \frac{e^{s_k} - e^{-s_k}}{e^{s_k} + e^{-s_k}}$$

With a slope parameter

$$z_k = \frac{e^{as_k} - e^{-as_k}}{e^{as_k} + e^{-as_k}}$$

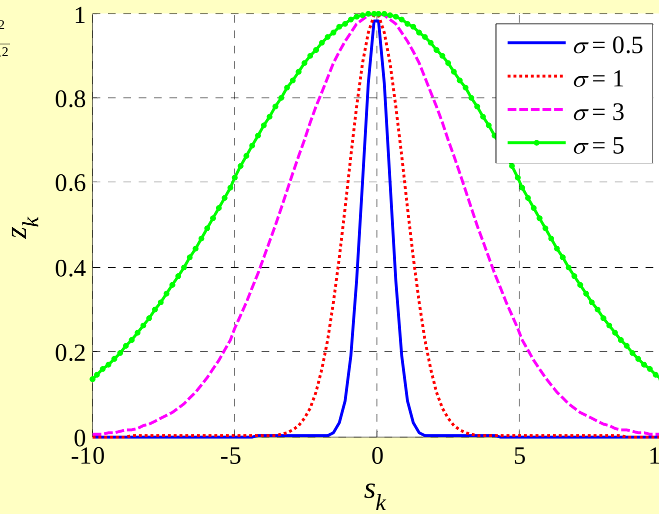


Typical Activation Functions (cont)

Gaussian function

$$z_k = \varphi_k(s_k) = e^{-\frac{s_k^2}{2\sigma^2}}$$

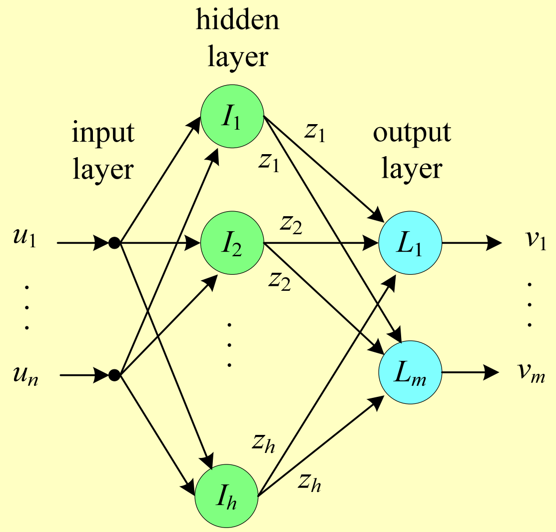
for some $\sigma > 0$



Some ANN Paradigms

- Multilayer Perceptrons
- Radial Basis Functions
- Recurrent Neural Networks

3-layer Perceptrons (3LP)



$$\boldsymbol{v} = \boldsymbol{b}^o + \boldsymbol{W}^o \boldsymbol{\Phi}(\boldsymbol{s})$$

$$\boldsymbol{s} = \boldsymbol{b}^h + \boldsymbol{W}^h \boldsymbol{u}$$

where

$$\boldsymbol{W}^o = [\boldsymbol{w}_1^{o\,T} \quad \cdots \quad \boldsymbol{w}_m^{o\,T}]^T$$

$$\boldsymbol{W}^h = [\boldsymbol{w}_1^{h\,T} \quad \cdots \quad \boldsymbol{w}_h^{h\,T}]^T$$

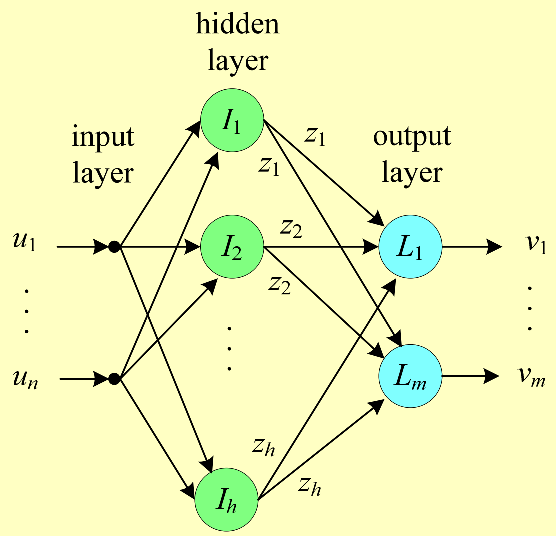
$$\boldsymbol{\Phi}(\boldsymbol{s}) = [\varphi(s_1) \quad \cdots \quad \varphi(s_h)]^T$$

$$\boldsymbol{b}^h = [b_1^h \quad b_2^h \quad \cdots \quad b_h^h]^T$$

$$\boldsymbol{b}^o = [b_1^o \quad b_2^o \quad \cdots \quad b_m^o]^T$$

$$\boldsymbol{v}, \boldsymbol{b}^o \in \Re^m \quad \boldsymbol{s}, \boldsymbol{b}^h, \boldsymbol{\Phi} \in \Re^h \quad \boldsymbol{W}^o \in \Re^{m\times h} \quad \boldsymbol{W}^h \in \Re^{n\times h}$$

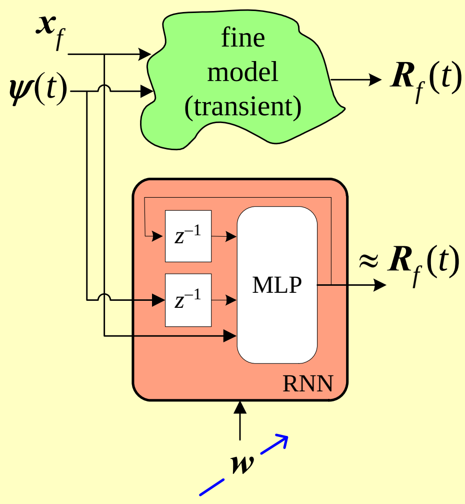
3-layer Perceptrons (3LP) (cont)



$$\boldsymbol{v} = \boldsymbol{N}(\boldsymbol{u},\boldsymbol{w})$$

where $\boldsymbol{w}$ contains $\boldsymbol{b}^o$, $\boldsymbol{b}^h$, $\boldsymbol{W}^o$ y $\boldsymbol{W}^h$

Neuromodels for Transient Domain



$\psi(t)$ input waveforms

$R_f(t)$ fine model output waveforms amplitudes

Critical issues for training RNNs:

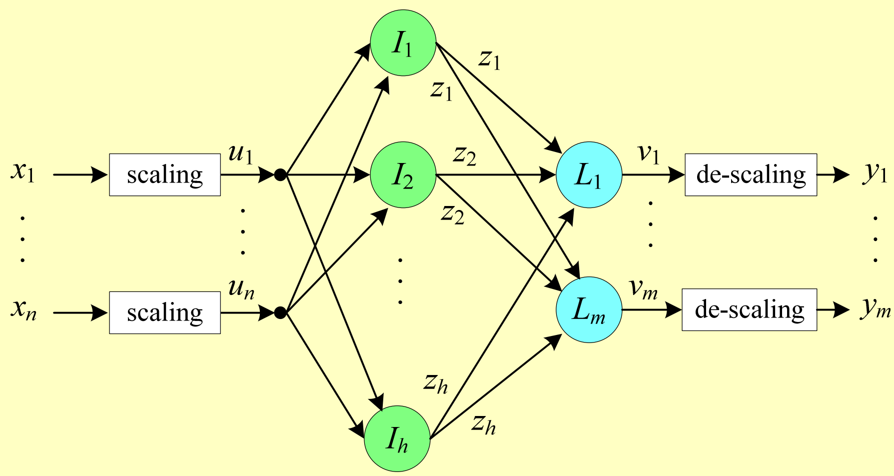
- sampling cycle
- number of unit-delay elements in each bank of delays

(Zhang et al., 2000, 2002)

Pre-Processing Training Data

- Scaling the input training data allows to control the relative importance of input parameters
- Scaling the input/output training data also defines the dynamic range of the actual training region
- Scaling improves convergence during ANN training

Pre-Processing Training Data (cont)



Pre-Processing Training Data (cont)

- Scaling between –1 and +1

$$u_i = -1 + \frac{2(x_i - x_{i\min})}{(x_{i\max} - x_{i\min})} \qquad i = 1, 2, \cdots, n$$

$$v_i = -1 + \frac{2(y_i - y_{i\min})}{(y_{i\max} - y_{i\min})} \qquad i = 1, 2, \cdots, m$$

- De-scaling

$$y_i = y_{i\min} + \frac{1}{2}(v_i + 1)(y_{i\max} - y_{i\min}) \qquad i = 1, 2, \cdots, m$$

Pre-Processing Training Data (cont)

- Scaling with zero mean and unitary standard deviation

$$u_i = \frac{x_i - x_i^{\text{mean}}}{x_i^{\text{SD}}} \qquad i = 1, 2, \cdots, n$$

$$v_i = \frac{y_i - y_i^{\text{mean}}}{y_i^{\text{SD}}} \qquad i = 1, 2, \cdots, m$$

- De-scaling

$$y_i = y_i^{\text{mean}} + v_i(y_i^{\text{SD}}) \qquad i = 1, 2, \cdots, m$$

Training ANNs

- Backpropagation algorithm
- Using optimization methods, for instance:
  - BFGS Quasi-Newton
  - Scaled Conjugate Gradient
  - Levenberg-Marquardt