Redes Neuronales

U1 Introducción

Héctor Maravillo

Section 1

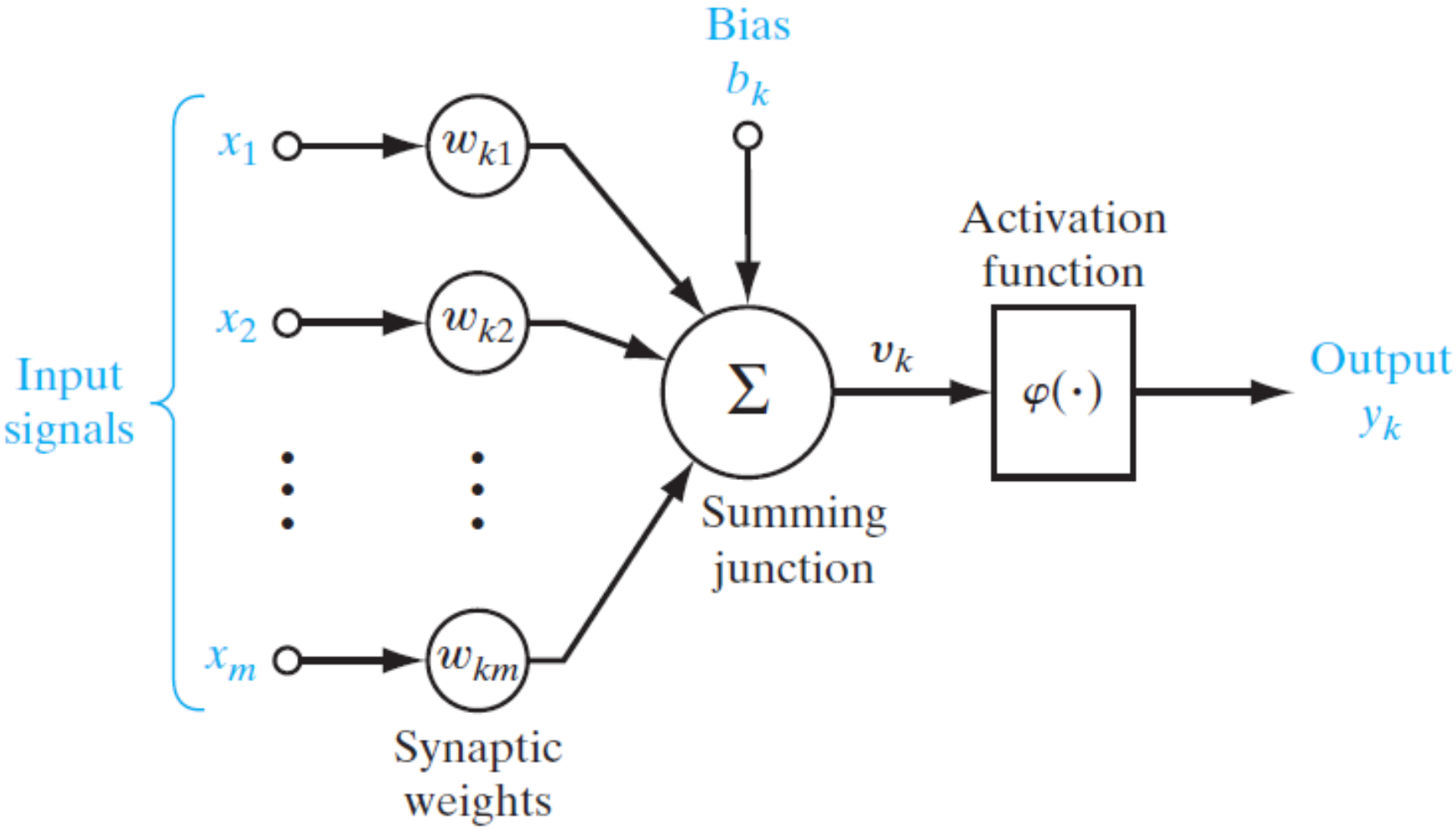
Architecture

Basic elements

- **Input signals** $(x_1, x_2, \ldots, x_n)$ are the signals or samples coming from the external environment.
- **Synaptic weights** $(w_1, w_2, \ldots, w_n)$ are the values used to weight each one of the input variables, which enables the quantification of their relevance with respect to the functionality of the neuron.
- **Linear aggregator** (Σ) combines all input signals weighted by the synaptic weights to produce an activation voltage.
- **Activation threshold** or **bias** (θ) is a variable used to specify the proper threshold that the result produced by the linear aggregator should have to generate a trigger value toward the neuron output.

- **Activation potential** (u) is the result produced by the difference between the linear aggregator and the activation threshold. If this value is positive $(u \geq \theta)$, then the neuron produces an excitatory potential; otherwise, it will be inhibitory.
- **Activation function** (g) whose goal is limiting the neuron output within a reasonable range of values
- **Output signal** (y) consists on the final value produced by the neuron given a particular set of input signals, and can also be used as input for other sequentially interconnected neurons.

Model of a neuron



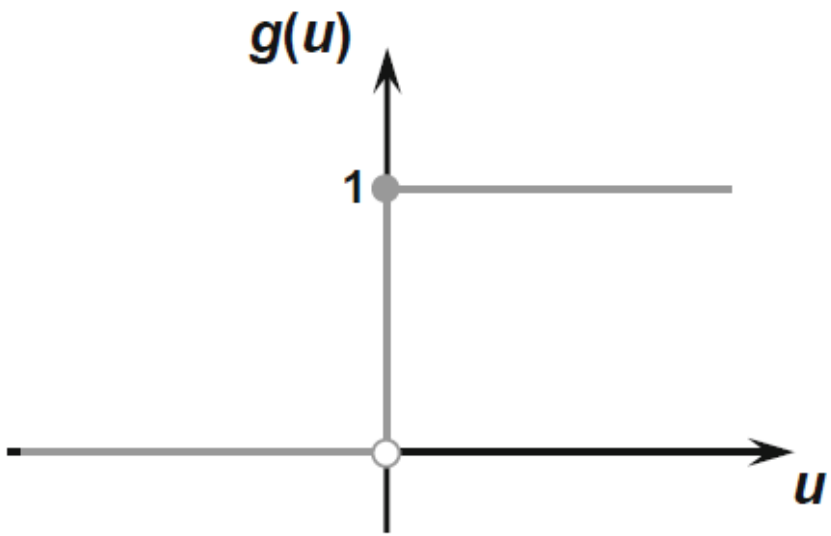
Section 2

Activation Functions

Partially differentiable activation function

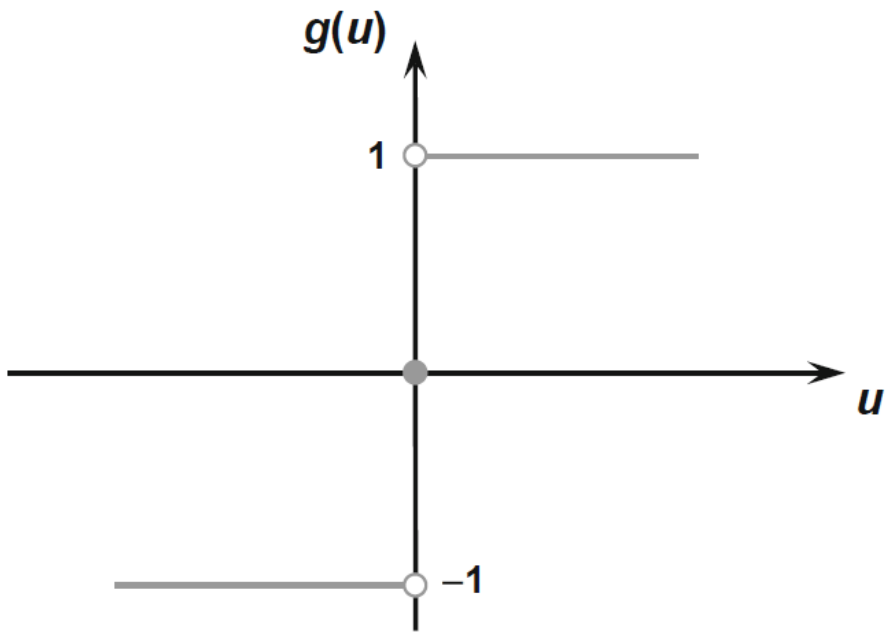
Step function The result will assume unitary positive values when the neuronal activation potential is grater than or equal to zero; otherwise, the result will be null.

$$g(u) = \begin{cases} 1, & \text{if } u \geq 0 \\ 0, & \text{if } u < 0 \end{cases}$$



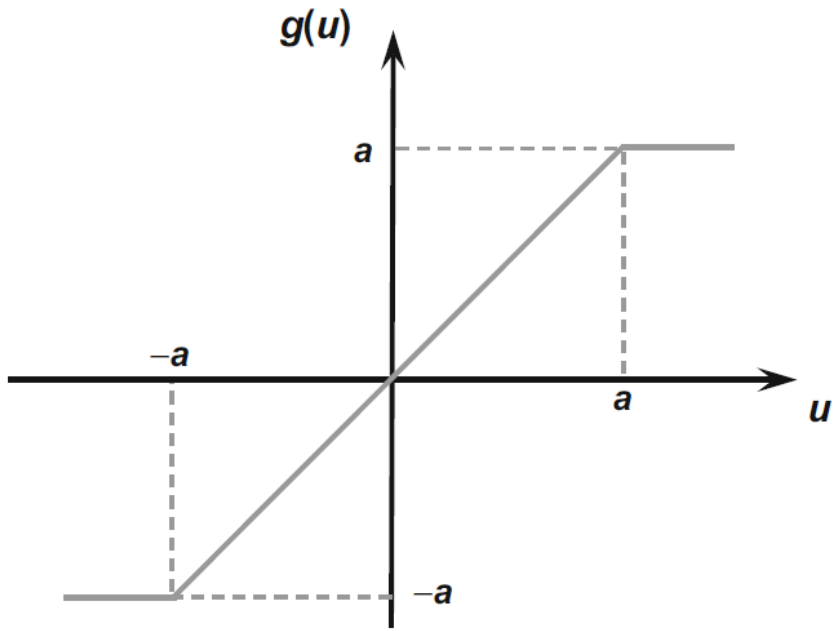
Bipolar Step function (signal function) The result will assume unitary positive value when the neuronal activation potential is grater than zero; null value when the potential is also null; and negative unitary values when the potential is less than zero.

$$g(u) = \begin{cases} 1, & \text{if } u > 0 \\ 0, & \text{if } u = 0 \\ -1, & \text{if } u < 0 \end{cases}$$



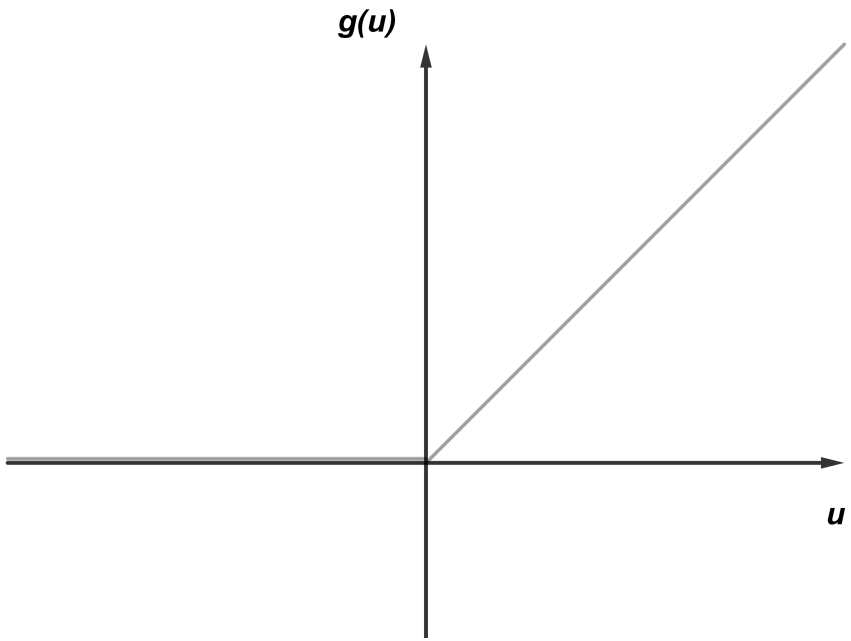
Symmetric ramp function The values returned by this function are equal to the values of the activation potential themselves when defined within the range $[-a, a]$, and limited to the limit values otherwise.

$$g(u) = \begin{cases} a, & \text{if } u > a \\ u, & \text{if } -a \leq u \leq a \\ -a, & \text{if } u < a \end{cases}$$



ReLU (Rectified Linear Unit) is an activation function defined as the non-negative part of its argument:

$$g(u) = \max(0, u) = \frac{u + |u|}{2}$$



ReLU is one of the most popular activation functions for artificial neural networks because:

- It mitigates the **vanishing gradient problem** compared to the sigmoid or the hyperbolic tangent.
- It's computationally **efficient**.
- It has **sparse activation**, meaning that for any given input, only a subset of neurons are activate which can improve computational efficiency.

Three generalizations of rectified linear units are based on using a **nonzero slope** α_i when $u < 0$:

$$g(u) = \max(0, u) + \alpha_i \min(0, u)$$

- **Absolute value rectification** fixes $\alpha_i = -1$ to obtain

  $$g(u) = |u|$$

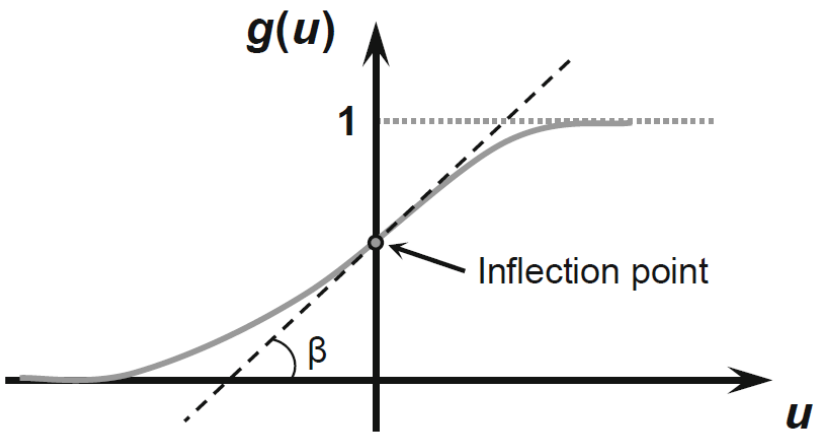
  It is used for object recognition from images where it makes sense to seek features that are invariant under a polarity reversal of the input illumination.
- A **Leaky ReLU** fixes α_i to a small value like 0.01.
- A **PReLU (Parametric ReLU)** treats α_i as a learnable parameter.

Fully differentiable activation function

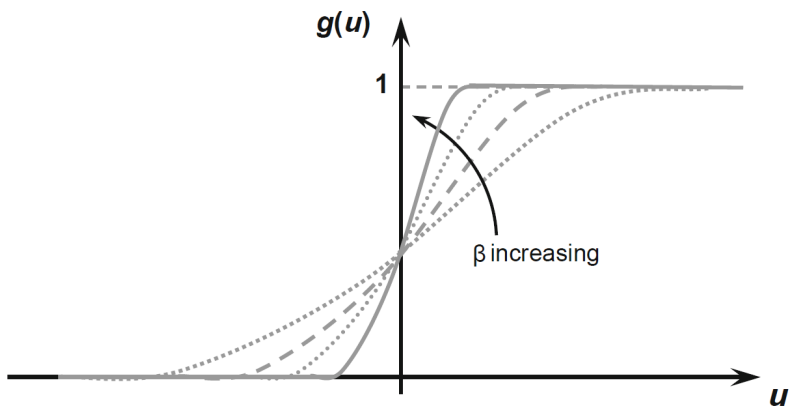
Logistic (sigmoid) function The output result produced by the logistic function will always assume real values in the range (0,1). Its mathematical expression is given by

$$g(u) = \frac{1}{1 + e^{-\beta u}}$$

where β is a real constant associated with the function slope in its inflection point.



(a) The logistic activation function



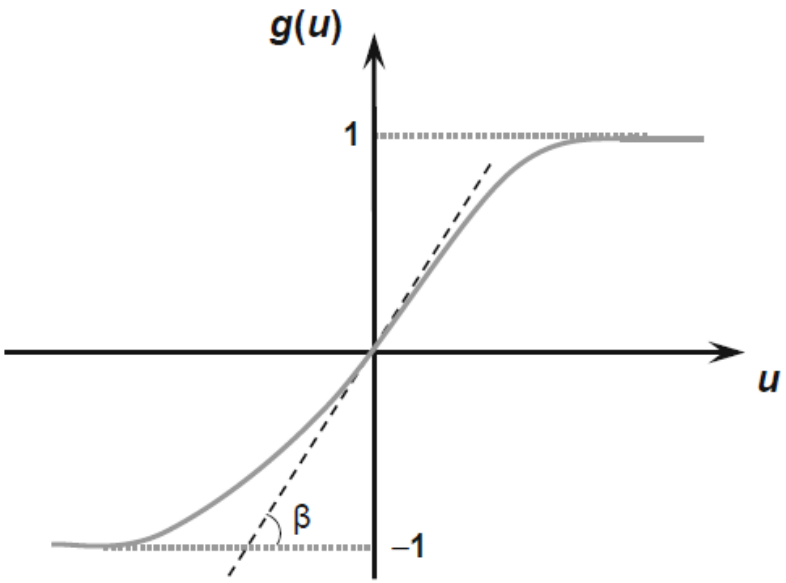
(b) Influence of the parameter β

Logistic (sigmoid) functionhas largely been replaced by the simpler and more easily trainable ReLU for most use in hidden layers.

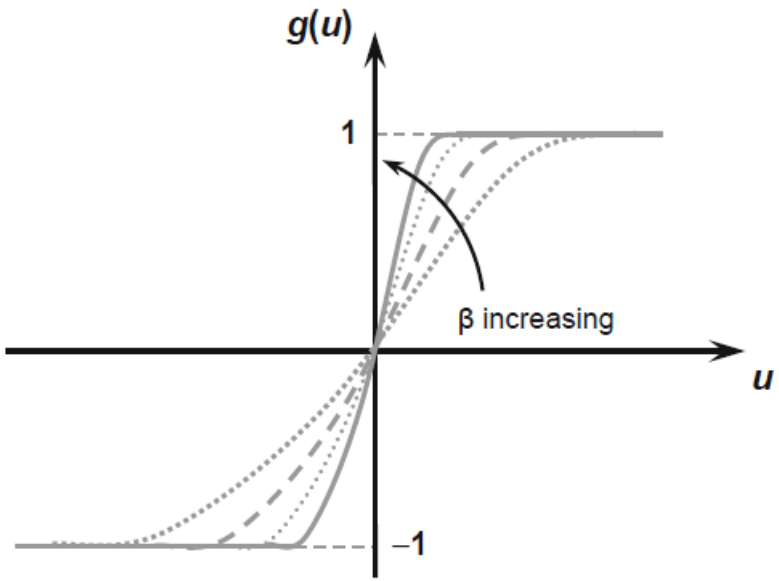
Much of this has to do with the fact that the sigmoid poses challenges for optimization since its **gradient vanishes** for **large positive** and **negative arguments**.

Hyperbolic tangent function The output result will always assume real values between -1 and 1, with the following mathematical expression:

$$g(u) = \tanh(u) = \frac{1 - e^{-\beta u}}{1 + e^{-\beta u}}$$



(a) The hyperbolic tanget function



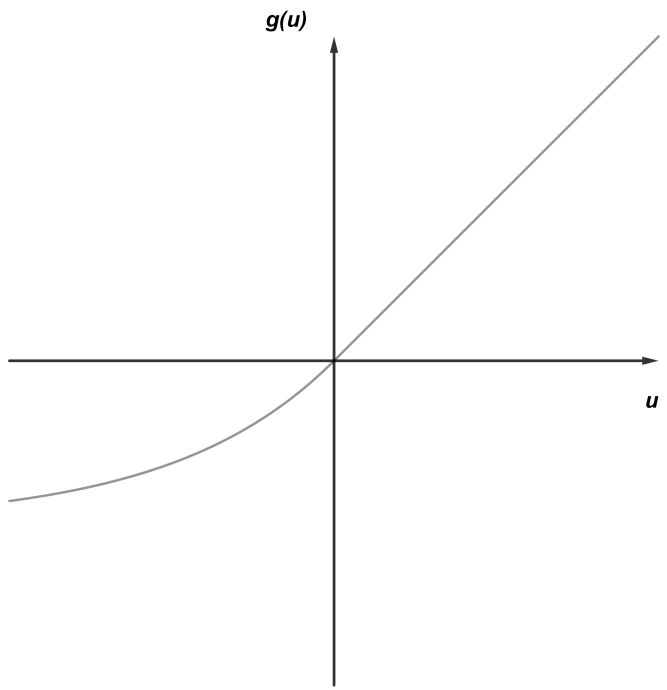
(b) Influence of the parameter β

[Note: The logistic and hyperbolic tangent functions belong to a family of functions called **sigmoidal**]

ELU (Exponential Linear Unit) is a continuous activation function that smoothly **allows negative outputs** for inputs less than zero, which helps bring the mean activations closer to zero and can improve the learning speed.

$$g(u) = \begin{cases} u & u > 0 \\ \alpha(e^u - 1) & u \leq 0 \end{cases}$$

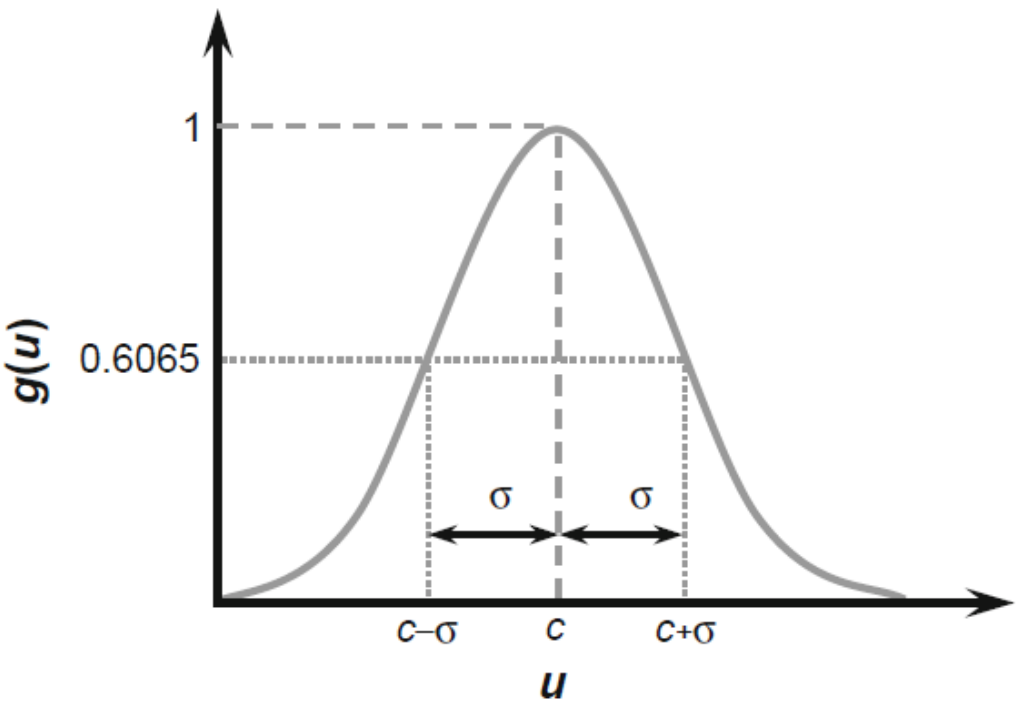
where $\alpha \geq 0$ is a hyperparameter.



Gaussian function The neuron output will produce results for those activation potential values u placed at the same distance from its center.

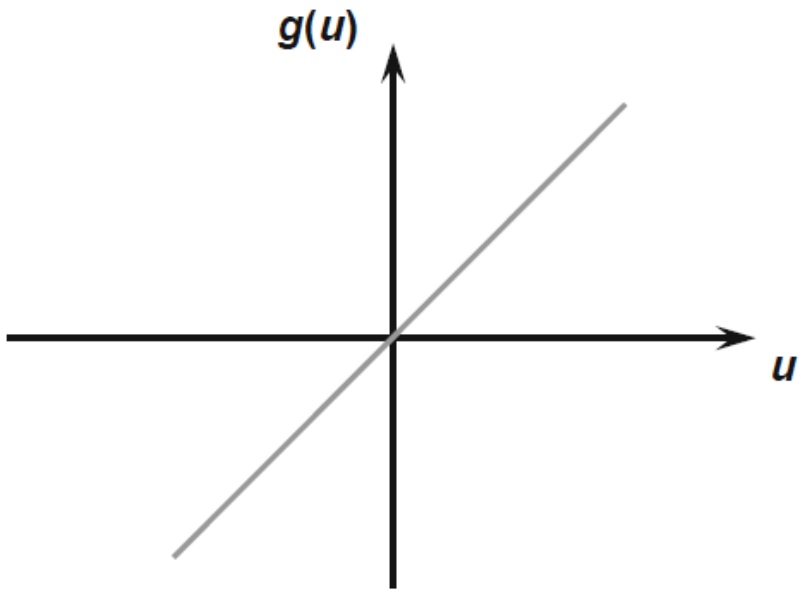
$$g(u) = e^{-\frac{(u-c)^2}{2\sigma^2}}$$

where c is the parameter that defines the **center** and σ denotes the associated **standard deviation**.



Linear function or **identity function** Produces output results equal to the activation potential.

$$g(u) = u$$



One application of the linear activation functions is in artificial neural networks performing universal curve fitting (function approximation).

Section 3

Network Architectures

Network Architectures

The manner in which the neurons of a neural network are **structured** is intimately linked with the **learning algorithm** used to train the network.

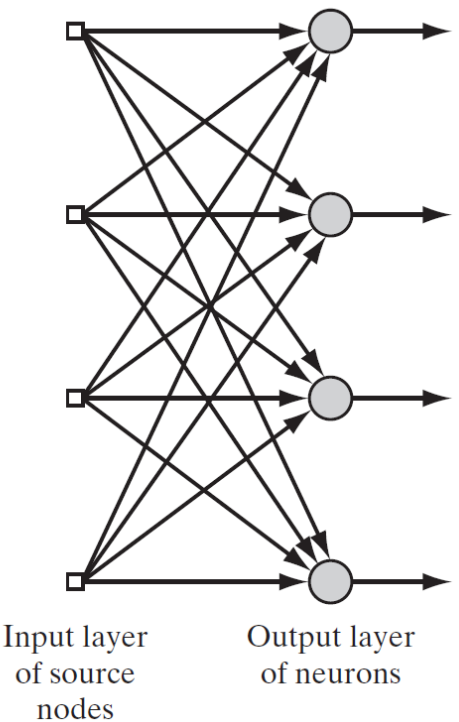
In general, we may identify three fundamentally different classes of network architectures:

- **Single-Layer Feedforward** networks.
- **Multilayer Feedforwar** networks.
- **Recurrent** networks.

Single-Layer

This artificial neural network has just one input layer and a **single neural layer**, which is also the **output layer** (i.e., the Perceptron and the ADALINE).

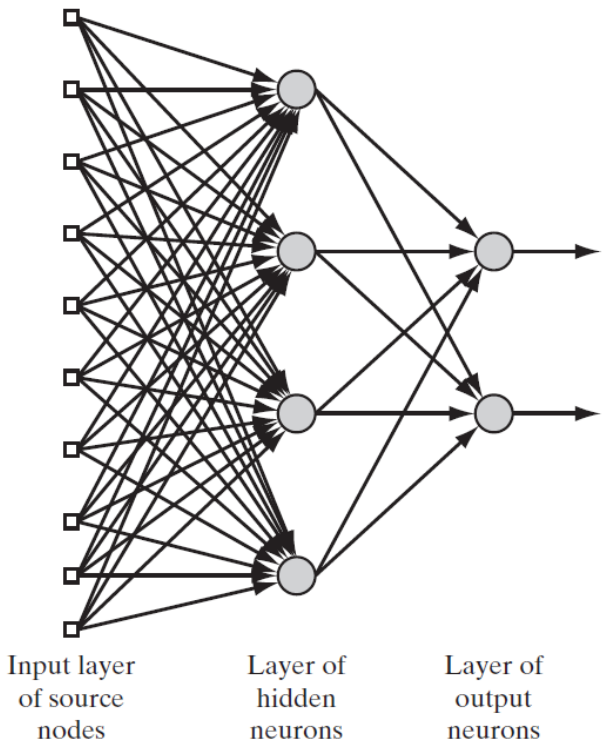
The information always flows in a single direction (thus, **unidirectional**), which is from the input layer to the output layer.



Multilayer

The second class of a feedforward neural network is distinguished by the presence of one or more **hidden layers**, whose computational nodes are correspondingly called **hidden neurons** or hidden units.

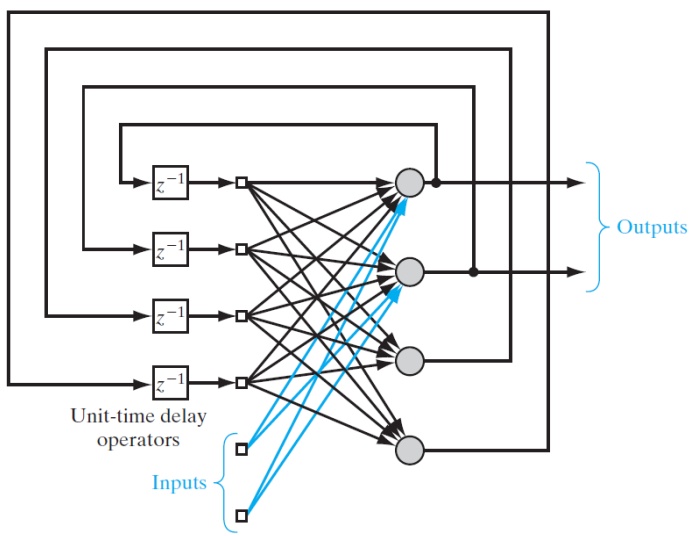
The term **hidden** refers to the fact that this part of the neural network is not seen directly from either the input or output of the network.



Recurrent Networks

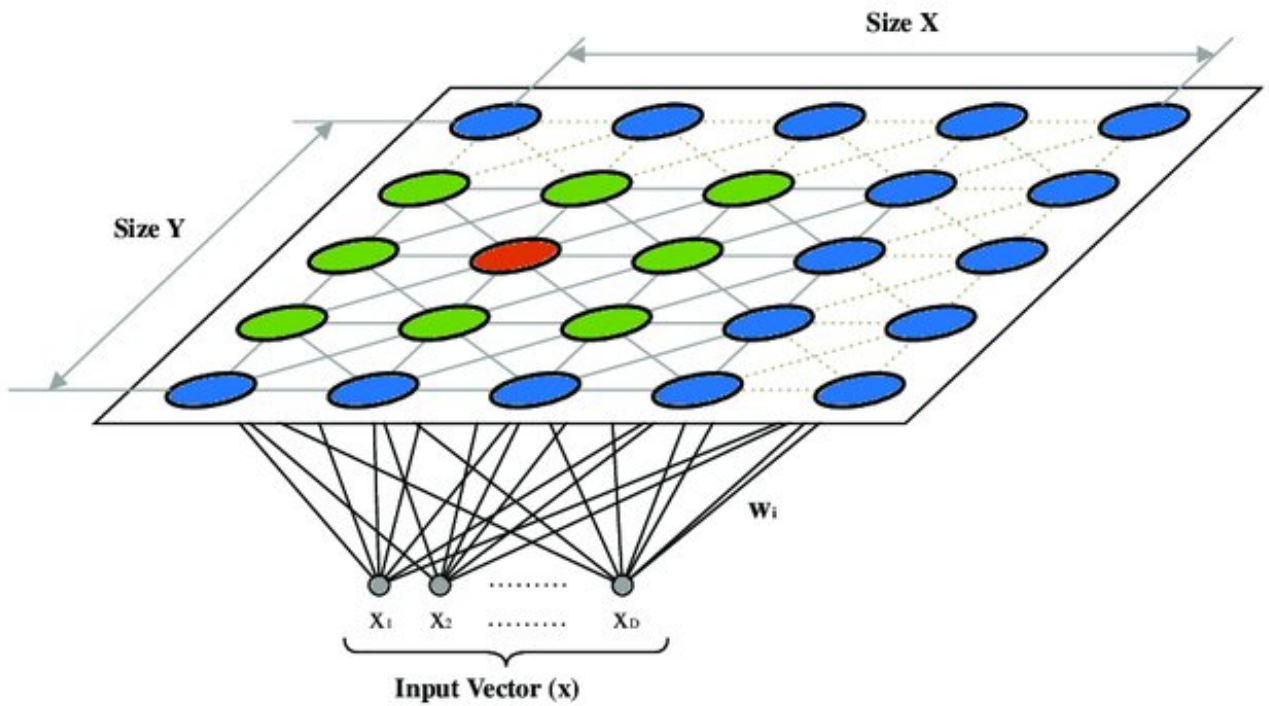
A **recurrent neural network** distinguishes itself from a feedforward neural network in that it has at least one **feedback loop** (for example, the Hopfield network).

The feedback feature qualifies these networks for **dynamics information processing**, meaning that they can be employed on **time-variant systems**, such as time series prediction, system identification and optimization, process control.



Mesh architectures

Its main feature reside in considering the spatial arrangement of neurons for pattern extraction purposes, that is, the **spatial localization** of the neurons is directly related to the process of adjusting their synaptic weights and threshold (for instance, Kohonen network).



Section 4

The McCulloch-Pitts Neuron

The McCulloch-Pitts neuron

The requeriments for **McCulloch-Pitts neuron** may be summarized as follows:

- The **activation** of a McCulloch-Pitts neuron is **binary**. That is, at any time step, the neuron either fires (has an activation of 1) or does not fire (has an activation of 0).
- McCulloch-Pitts neurons are connected by **directed, weighted** paths.
- A connection path is **excitatory** if the weight on the path is positive; otherwise it is inhibitory.
- All excitatory connection into a particular neuron have the **same weights**.

The McCulloch-Pitts neuron

- Each neuron has a **fixed threshold** such tat if the net input to the neuron is greater than the threshold, the neuron fires.
- The threshold is set so that inhibition is **absolute**. That is, any nonzero inhibitory input will prevent the neuron from firing.
- It takes one time step for a signal to pass over one connection link.

The McCulloch-Pitts Architecture

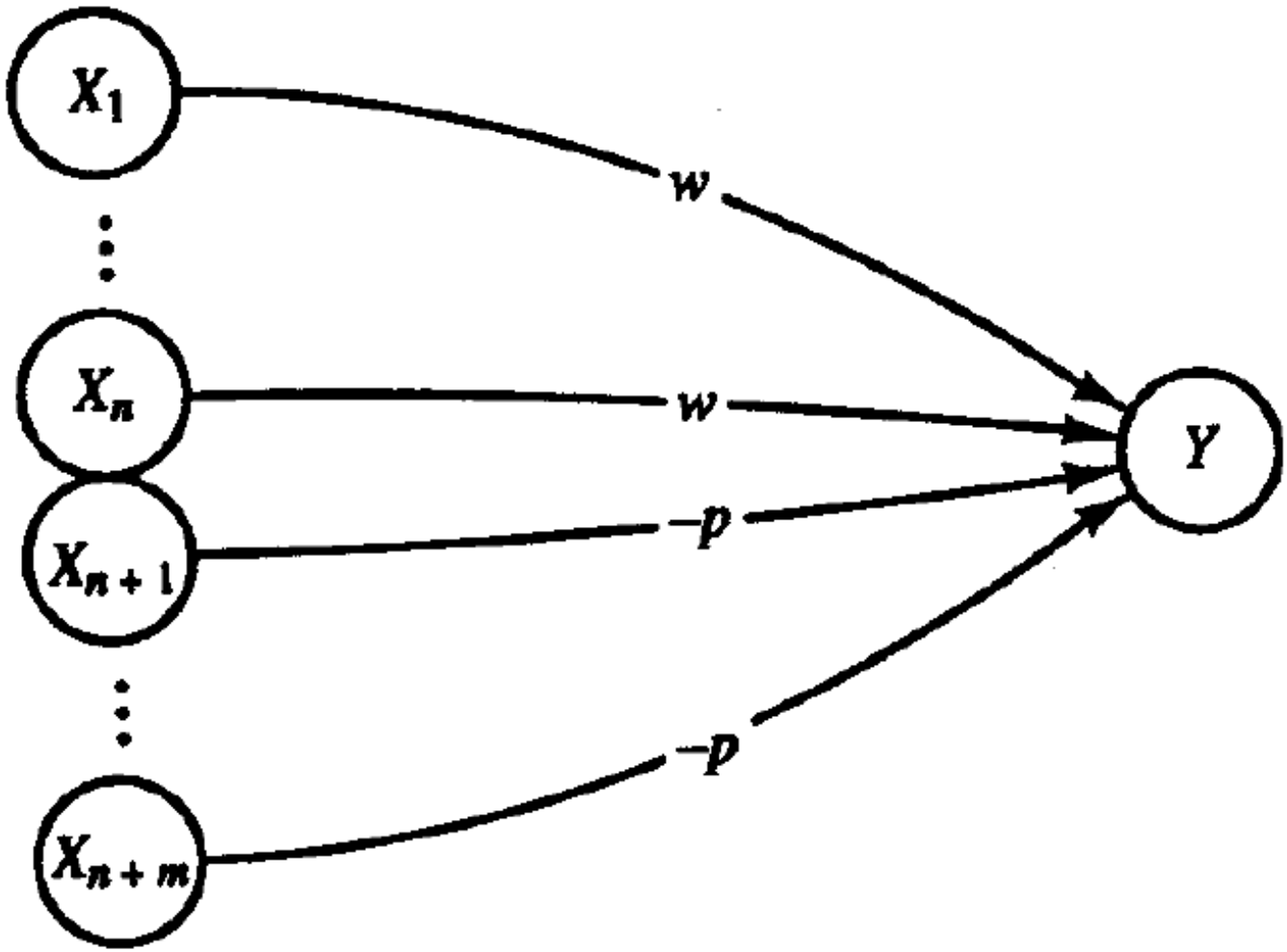
In general, a **McCulloch-Pitts neuron** Y may receive signals from any number of other neurons.

Each connection path is either **excitatory**, with $w > 0$, or **inhibitory**, with weight $-p$, $(p > 0)$.

For convenience, we assume there are n units $X_1, \ldots, X_n$, which send **excitatory signals** to unit Y, and m units, $X_{n+1}, \ldots, X_{n+m}$, which send **inhibitory signals**.

Although all excitatory weights coming into any particular unit must be the same, the weights coming into one unit, say, Y_1, do not have to be the same as the weights coming into another unit, say Y_2.

The McCulloch-Pitts Architecture



The McCulloch-Pitts Architecture

The **activation function** for unit T is

$$f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} \geq \theta \\ 0 & \text{if } y_{in} < \theta \end{cases}$$

where y_{in} is the total input signal received and θ is the threshold.

The McCulloch-Pitts algorithm

The weights for a McCulloch-Pitts neuron are set, together with the threshold for the neuron's activation function, so that the neuron will perform a **simple logic function**.

Rather than learning, analysis was used to determine the weights and threshold.

Using these simple neurons as building blocks, we can **model any function** or phenomenon that can be represented as a **logic function**.

References

- Fausset, L. *Fundamentals of Neural Networks. Architectures, Algorithms and Applications*, Pearson Education, 1994.
- Nunes da Silva, I., Hernane D., Andrade, R., Bartocci, L., and dos Reis, S. *Artificial Neural Networks. A Practical Course*, Springer, 2017.
- Haykin, S. *Neural Networks and Learning Machines*, 3rd ed., Pearson, 2009.