Artificial Neural Networks

Chapter 1: The Artificial Neuron

Prof. D.Sc. BARSEKH-ONJI Aboud

Facultad de Ingeniería
Universidad Anáhuac México

January 15, 2026

Agenda

Introduction to the Artificial Neuron

Biological Inspiration

The Artificial Neuron (AN) is the fundamental building block of Artificial Neural Networks (ANNs), inspired by the biological neuron. It receives input signals, processes them, and produces an output signal if the accumulated input reaches a certain threshold.

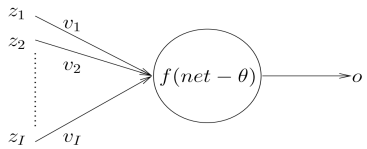


Figure 1: Schematic representation of an Artificial Neuron.

Components of an Artificial Neuron

- **Input Signals ($z_1, z_2, \ldots, z_I$):** Represent data or the output from other neurons.
- **Weights ($v_1, v_2, \ldots, v_I$):** Represent the synaptic strength of connections. Positive weights are excitatory, negative are inhibitory.
- **Combination Function (Net Input):** Aggregates the weighted inputs (usually summation).
- **Activation Function (f_{AN}):** Defines the output based on the net input.
- **Bias (θ):** A threshold value that shifts the activation function.

Agenda

Calculating the Net Input Signal

The net input signal (*net*) combines all incoming signals and weights. There are two main types:

Summation Unit (SU)

The most common type, calculating the weighted sum of inputs minus the bias.

$$net = \sum_{i=1}^{I} z_i v_i - \theta \qquad (1)$$

Product Unit (PU)

Multiplies the inputs raised to the power of their weights.

$$net = \prod_{i=1}^{I} z_i^{v_i} \qquad (2)$$

Agenda

Activation Functions

The activation function $f_{AN}(net)$ determines the output o of the neuron. It defines the relationship between the internal state and the output signal.
Common characteristics required:

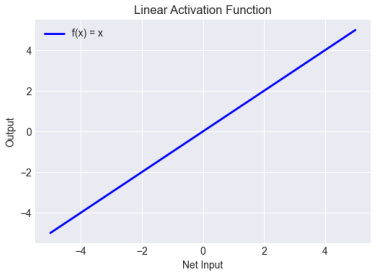
- Non-linearity (essential for multi-layer networks).
- Differentiability (required for Gradient Descent learning).
- Boundedness (often desirable).

Linear and Step Functions

Linear Function

$f(net) = \gamma \cdot net$

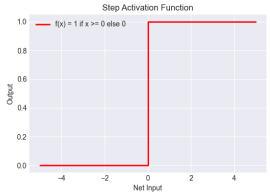
Simple identity, no non-linearity.



Step Function

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

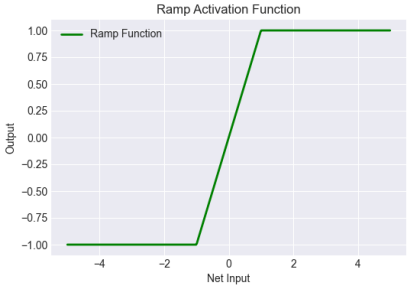
Used in the original Perceptron. Not differentiable.



Ramp and Sigmoid Functions

Ramp Function

Combination of linear and step, bounded.

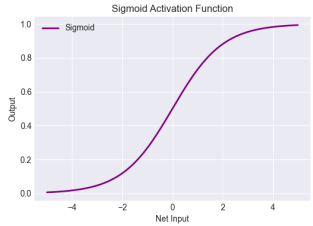


Sigmoid Function

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

Continuous, differentiable, S-shaped.

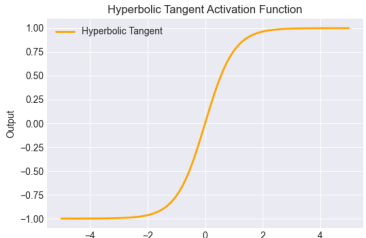
Crucial for Backpropagation.



Hyperbolic Tangent and Gaussian

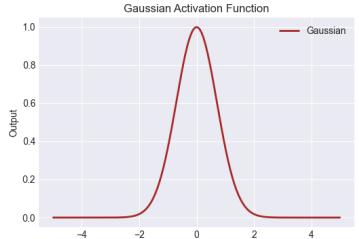
Hyperbolic Tangent (tanh)

$f(net) = \tanh(\lambda net)$
Like Sigmoid but ranges from -1 to 1.
Zero-centered.



Gaussian Function

$f(net) = e^{-net^2}$
Used in Radial Basis Function (RBF) networks. Localized response.



Agenda

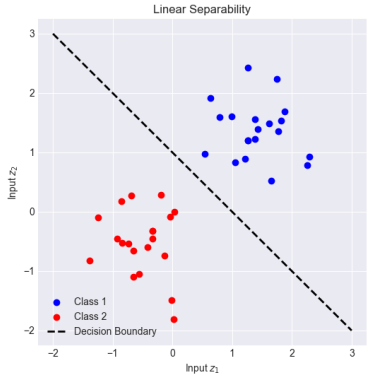
Artificial Neuron Geometry

Hyperplane Decision Boundary

An Artificial Neuron with a step activation function acts as a linear classifier. It divides the input space into two regions separated by a **decision boundary** (hyperplane).
The boundary is defined where the net input is zero:

$$\sum_{i=1}^{I} z_i v_i - \theta = 0 \tag{3}$$

Artificial Neuron Geometry



Agenda

Learning in Artificial Neurons

Learning involves adjusting the weights (v_i) and bias (θ) to minimize the error between the actual output and the desired target.

Augmented Vectors

To simplify learning, the bias is treated as a weight v_{I+1} connected to an input $z_{I+1} = -1$.

$$net = \sum_{i=1}^{I+1} z_i v_i = \mathbf{Z}^T \mathbf{V} \tag{4}$$

Gradient Descent Learning Rule

Gradient Descent

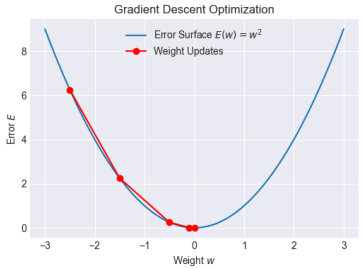
Minimizes the Sum-Squared Error (SSE) by moving weights in the opposite direction of the gradient.
Weight update rule:

$$\Delta v_i = \eta(t_p - o_p)f'(net)z_{i,p} \tag{5}$$

Where η is the learning rate, t_p is target, and o_p is actual output.

Gradient Descent Learning Rule

This requires the activation function f to be differentiable (e.g., Sigmoid).



Other Learning Rules

Widrow-Hoff (LMS) Rule

Also known as the Delta Rule. Assumes $f(net) = net$ (linear) for the error calculation.

$$v_i(t+1) = v_i(t) + 2\eta(t_p - o_p)z_{i,p} \tag{6}$$

Error-Correction Learning

Updates weights only when the neuron makes a mistake (used in Perceptron).

- If output is correct ($o_p = t_p$), no change.
- If output is 0 but should be 1, increase weights.
- If output is 1 but should be 0, decrease weights.

Agenda

Summary

- The Artificial Neuron is a mathematical abstraction of the biological neuron.
- It computes a weighted sum of inputs and applies a non-linear activation function.
- **Activation functions** (Sigmoid, Tanh, ReLU) introduce non-linearity, enabling the learning of complex patterns.
- **Learning rules** (Gradient Descent, Widrow-Hoff) allow the neuron to adapt its weights to minimize error.
- Single neurons define linear decision boundaries (Hyperplanes).