ARTIFICIAL NEURAL NETWORK

An **artificial neural network (ANN)** is a subfield of Artificial Intelligence where it attempts to mimic the network of neurons that makes up a human brain so that computers will have the option to understand things and make decisions in a human-like manner. The ANN is designed by programming computers to behave simply like interconnected brain cells.

An ANN consists of an **input layer**, **hidden layers** of **nodes** (or **neurons**, or **perceptrons**), and an **output layer**. The first layer receives raw input, it is processed by multiple hidden layers, and the last layer produces the result.

Historical Note: The oldest type of neural network, known as **Perceptron**, was introduced by Frank Rosenblatt (1928-1971) in 1958.

Rosenblatt, F. (1958). "The perceptron: A probabilistic model for information storage and organization in the brain." *Psychological Review*, 65(6), 386–408.

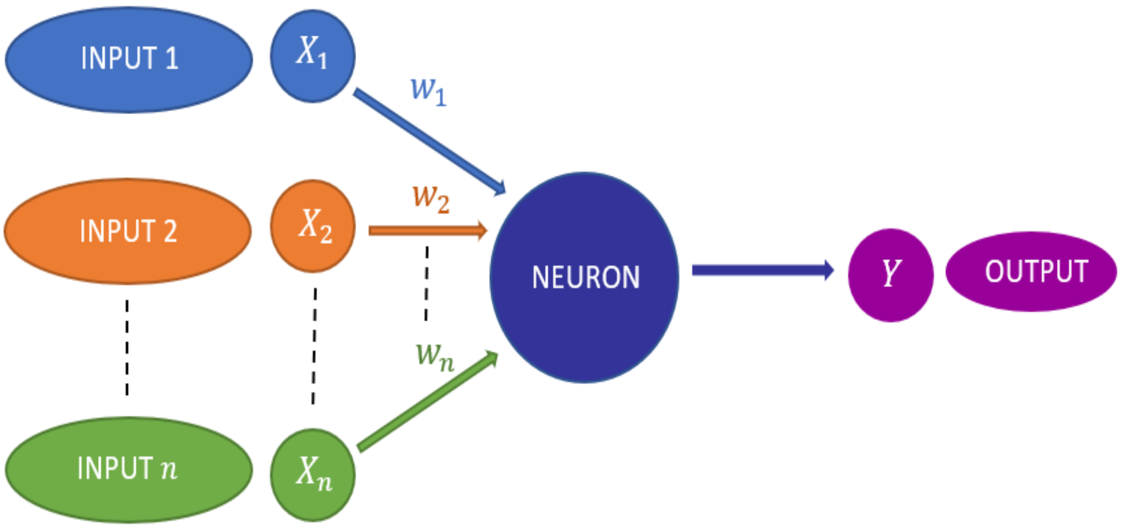
CORNELL CHRONICLE

Division of Rare and Manuscript Collections

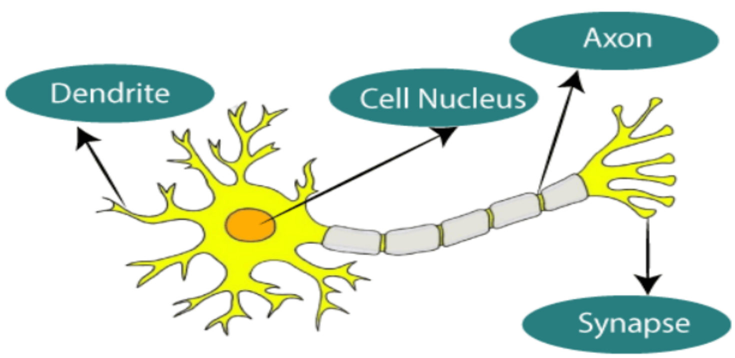
Frank Rosenblatt '50, Ph.D. '56, works on the "perceptron" – what he described as the first machine "capable of having an original idea."

Professor's perceptron paved the way for AI – 60 years too soon

A typical ANN looks something like this:



A typical diagram of a biological neural network in the brain looks like this:

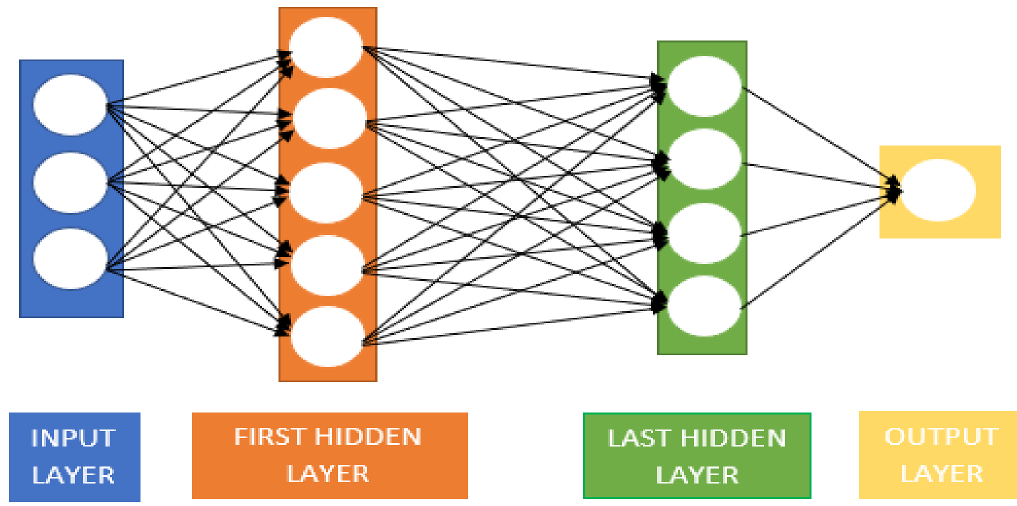


Dendrites from biological neural networks represent inputs in ANN, cell nucleus represents nodes, synapse represents weights, and axon represents output.

Glossary

• **Dendrite** is a short-branched extension of a nerve cell, along which impulses received from other cells at synapses are transmitted to the cell body.
• **Synapse** is a junction between two nerve cells, consisting of a minute gap across which impulses pass by diffusion of a neurotransmitter.
• **Axon** is a long threadlike part of a nerve cell along which impulses are conducted from the cell body to other cells.

To understand the concept of the architecture of an ANN, we have to understand what a neural network consists of. In order to define a neural network that consists of a large number of artificial neurons, which are nodes arranged in a sequence of layers. Let us look at three types of layers available in an ANN: input layer, hidden layer, and output layer.



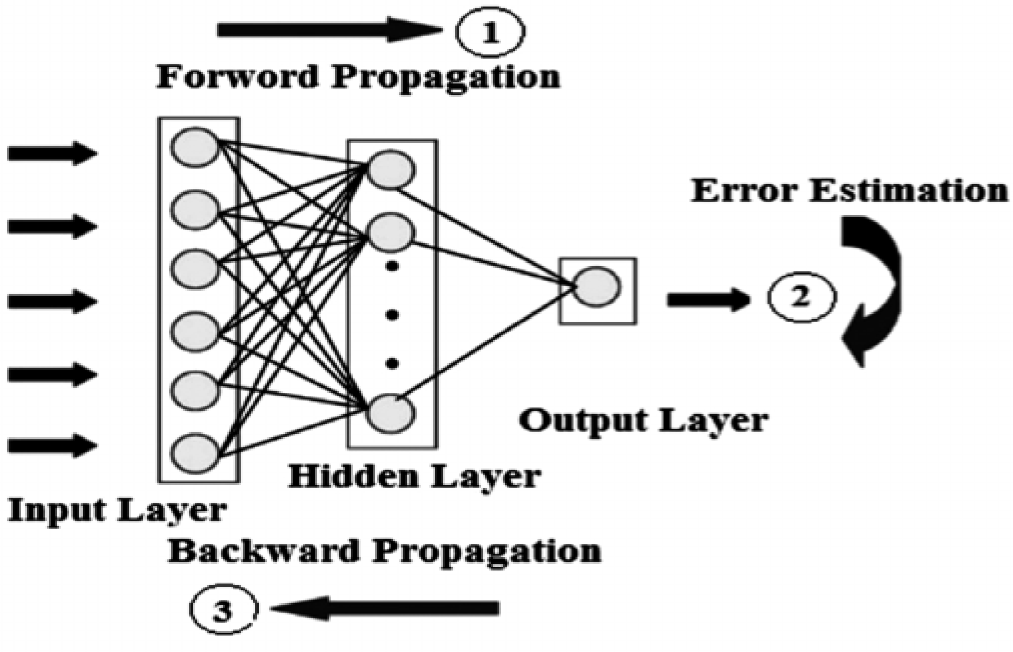
Input Layer accepts inputs provided by a programmer. The **input features** (or the predictor variables) can be categorical or numeric.

Hidden Layer performs all the calculations to find hidden features and patterns.

Output Layer consists of the output variable (or response variable). For regression ANNs, the output variable is numeric; for binary ANN, the output variable is binary, and for multinomial ANN, the output variable assumes multi-class values.

In an ANN, the input goes through a series of transformations using the hidden layer, which finally results in the output expressed as a linear combination of weighted input features with a bias term included.

It determines the weighted total that is passed as an input to an **activation function** to produce the output. Activation functions choose whether a node should fire or not. Only those who are fired make it to the output layer. This process is called **feed forward** (or **forward propagation**). After producing the output, an error (or loss) is calculated and a correction is sent back to the network. This process is known as **back propagation** (or **backward propagation**).



Historical Note. ANN with back propagation was introduced in Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). "Learning representations by back-propagating errors". *Nature*, 323(6088), 533–536. Most of the ANN applications in the literature utilize multi-layer feed-forward with a back propagation learning algorithm.

An **epoch** is a complete cycle through the full training set when building an ANN. An **iteration** is the number of steps through partitioned packets of the training data, needed to complete one epoch.

Learning Algorithm

An ANN starts with a set of initial weights and then gradually modifies the weights during the training cycle to settle down to a set of weights capable of realizing the input-output mapping with a minimum error.

Denote by $\mathbf{x_i} = (x_{i1}, \dots, x_{ik})'$, $i = 1, \dots, n$, the set of vectors of input variables (predictor variables), and let $\hat{\mathbf{y}} = (\hat{y}_1, \dots, \hat{y}_n)$ be the output vector. Also, suppose there is one hidden layer with m neurons $h_1, \dots, h_m$. The response of the hidden layer for the ith individual is the vector $\mathbf{h_i} = (h_{i1}, \dots, h_{im})'$. An ANN produces outputs governed by the relations:

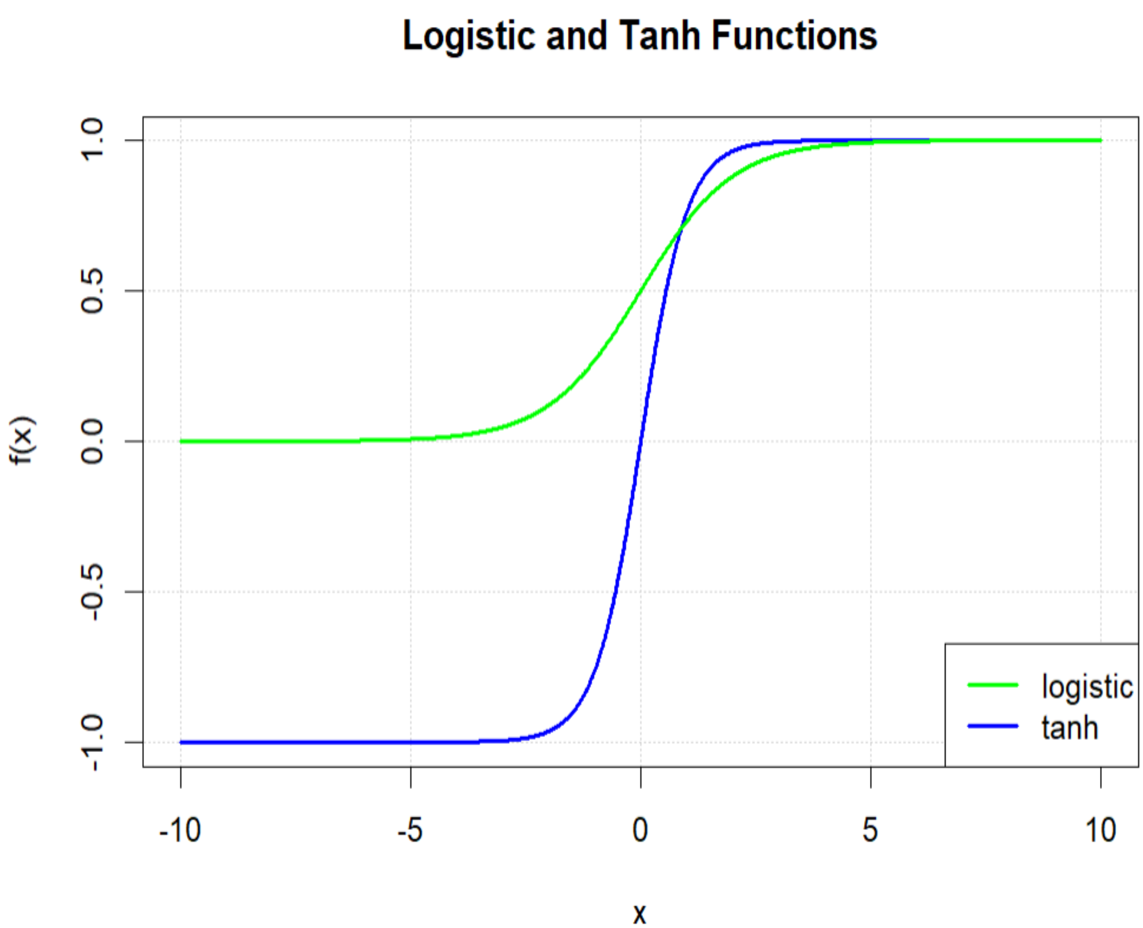
$$\mathbf{h_i} = f\big(\mathbf{W_h}\,\mathbf{x_i} + \mathbf{b_i}\big), \quad \text{and} \quad \hat{y}_i = f(\mathbf{W_i^*}\,\mathbf{h_i} + b_i^*),$$

where f is the activation function,

$$\mathbf{W_h} = \begin{bmatrix} w_{11} & \dots & w_{1k} \\ \dots & \dots & \dots \\ w_{m1} & \dots & w_{mk} \end{bmatrix}$$

is the hidden layer weight matrix, $\mathbf{W_i^*} = (w_{i1}^*, \dots, w_{im}^*)$ is the vector of output weights for individual i, $\mathbf{b_i} = (b_{i1}, \dots, b_{im})'$ is the hidden layer bias vector for individual i, and b_i^* is the output layer bias for individual i.

The activation functions that are used in SAS, R, and Python are (defined for $x \in \mathbb{R}$) **logistic** (or **sigmoid**) $f(x) = \dfrac{\exp(x)}{1+\exp(x)}$, $-\infty < x \infty$, and **hyperbolic tangent** (or **tanh**) $f(x) = \tanh(x) = \dfrac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$, $-\infty < x < \infty$. Both functions are illustrated below. Note that the logistic functions ranges between 0 and 1, whereas the tanh function ranges between -1 and 1. Also, the first function at zero is 0.5, while the second function is 0.



The loss functions used to compute errors in the back propagation algorithm are: mean squared error for regression and cross-entropy for classification.

The method of **steepest descent** is used to update the weights. For example, for the mean squared error loss function, the loss function is $L = \dfrac{1}{n}\sum_{i=1}^{n}\Big(y_i - f(\mathbf{W_i^*}\,\mathbf{h_i} + b_i^*)\Big)^2$. The weights are updated according to the recursive relation $w_{ij}^*(new) = w_{ij}^*(old) - \lambda \dfrac{\partial L}{\partial w_{ij}^*}, j = 1, \dots, m,$, where λ is referred to as **learning rate**. The same algorithm applies to the weights in the hidden layers W_h.

ANN Binary Classifier

For an ANN binary classifier, the loss function is the average cross-entropy across all data points

$$L \;=\; \frac{1}{n}\sum_{i=1}^{n}\big[y_i \ln \hat{y}_i \;+\; (1-y_i)\ln(1-\hat{y}_i)\big]$$

$$= \frac{1}{n}\sum_{i=1}^{n}\Big[y_i \ln\big(f(\mathbf{W_i^*}\,\mathbf{h_i} + b_i^*)\big) \;+\; (1-y_i)\ln\big(1 - f(\mathbf{W_i^*}\,\mathbf{h_i} + b_i^*)\big)\Big].$$

ANN Multinomial Classifier

The loss function used in multinomial classification is the multinomial cross-entropy function defined as $E = -\sum_{i=1}^{k}\big[p_i \ln p_i\big]$ where p_i is the proportion of observations in class i, $i = 1, ..., k$.