GATE WAY TO ML

ANN

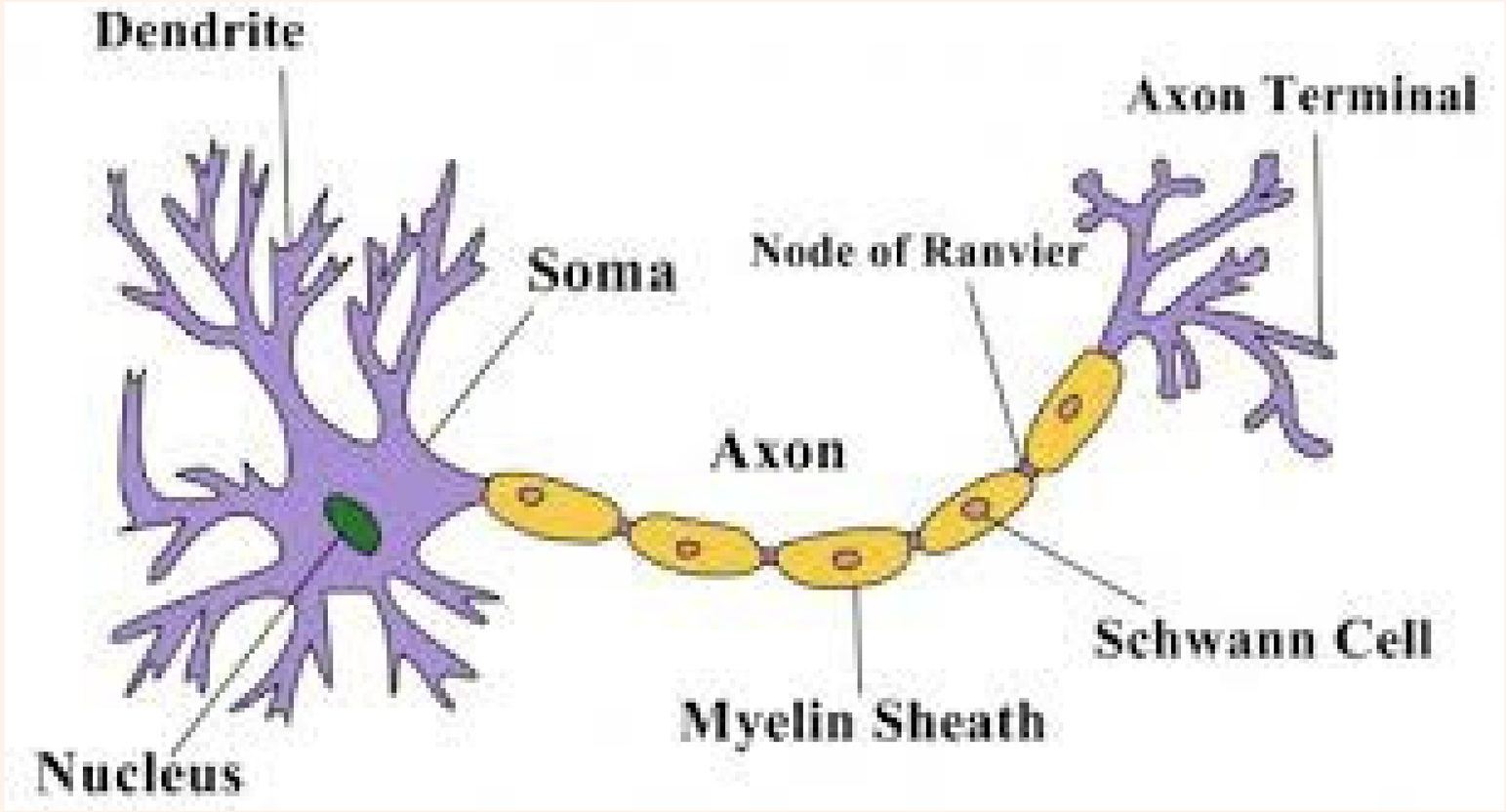
Artificial Neural Network

A neural network is a powerful data modeling tool that is able to capture and represent complex input/output relationships.

The motivation for the development of neural network technology stemmed from the desire to develop an artificial system that could perform "intelligent" tasks similar to those performed by the human brain.

Artificial Neural Network

BIOLOGICAL NEURON



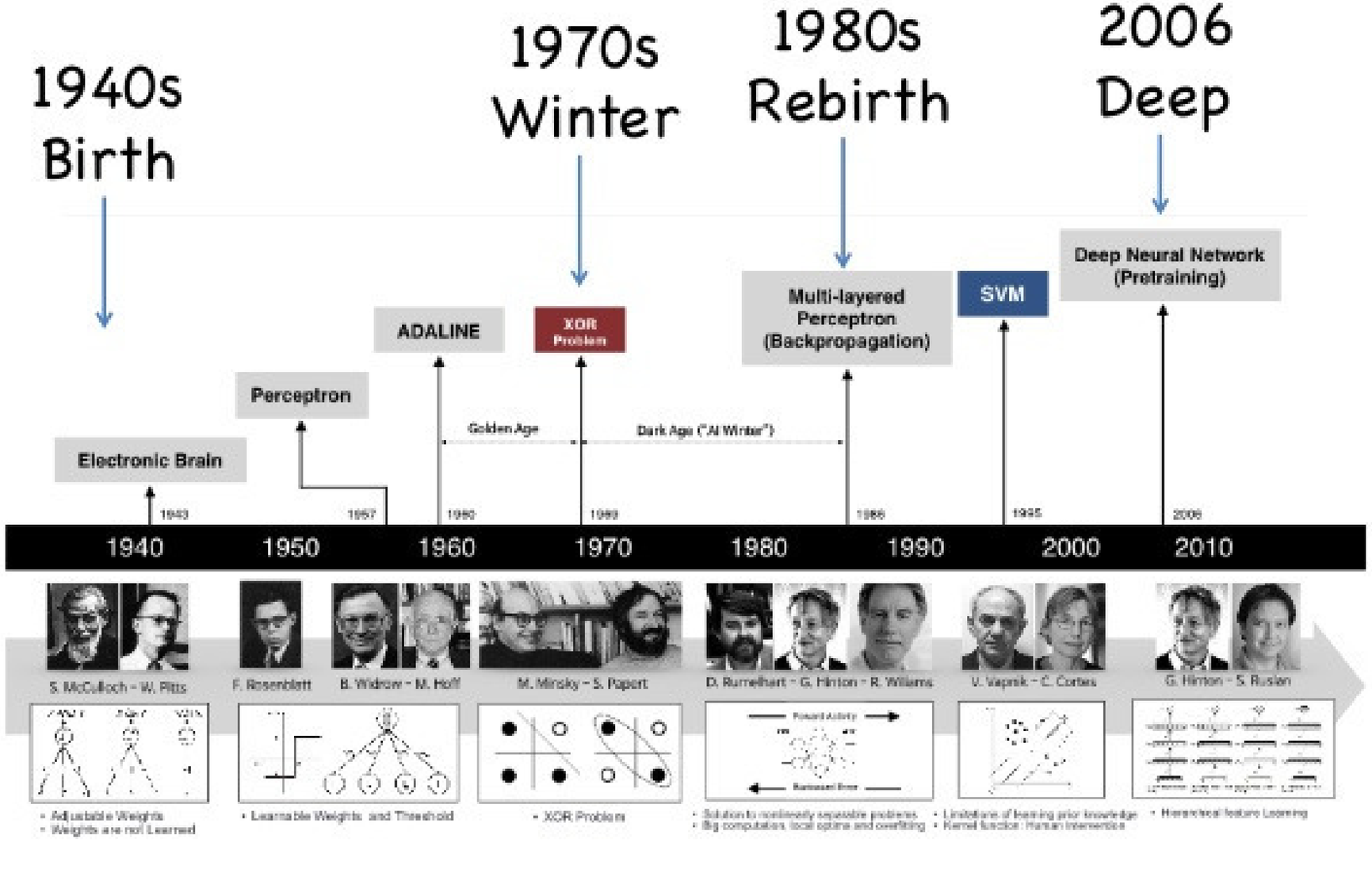
Artificial Neural Network

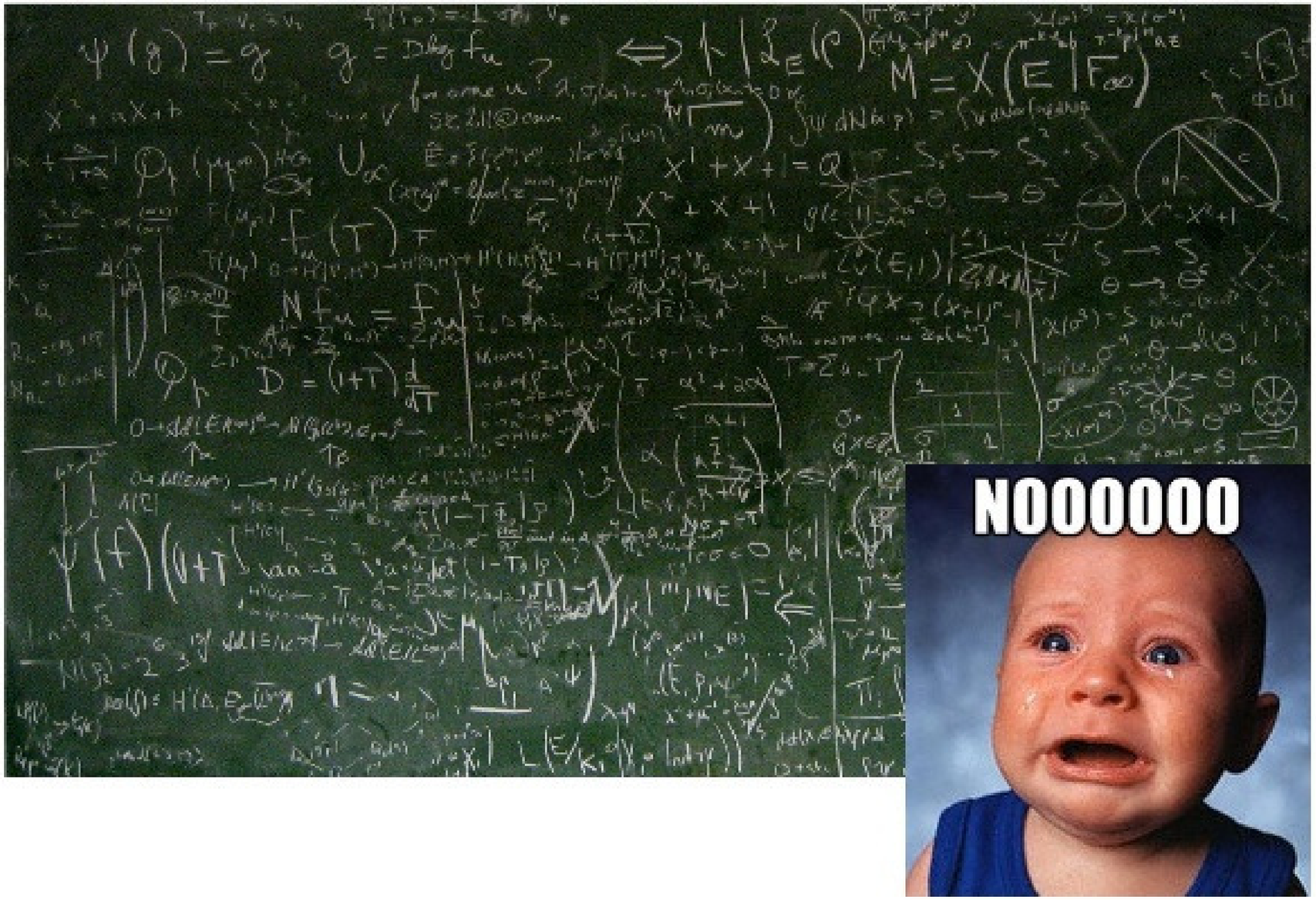
Neural networks resemble the human brain in the following two ways:

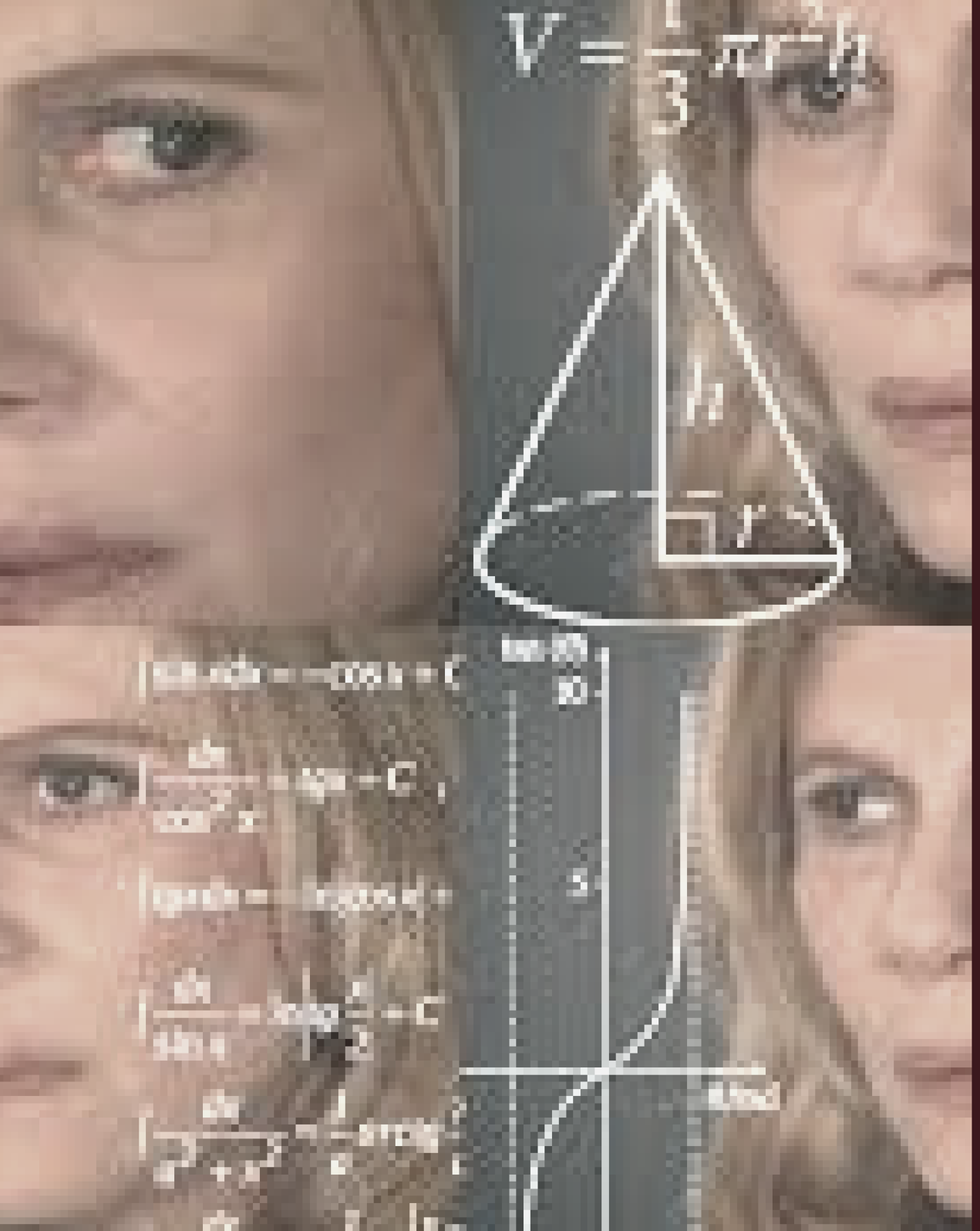
1.A neural network acquires knowledge through learning.

2.A neural network's knowledge is stored within inter-neuron connection strengths known as synaptic weights.

History


1940s
Birth
1970s
Winter
1980s
Rebirth
2006
Deep
Electronic Brain
Perceptron
ADALINE
XOR
Problem
Multi-layered
Perceptron
(Backpropagation)
SVM
Deep Neural Network
(Pretraining)
Golden Age
Dark Age ("AI Winter")
1943
1957
1960
1969
1986
1995
2006
1940
1950
1960
1970
1980
1990
2000
2010
S. McCulloch - W. Pitts
F. Rosenblatt
B. Widrow - M. Hoff
M. Minsky - S. Papert
D. Rumelhart - G. Hinton - R. Williams
V. Vapnik - C. Cortes
G. Hinton - S. Ruslan
• Adjustable Weights
• Weights are not Learned
• Learnable Weights and Threshold
• XOR Problem
• Solution to nonlinearly separable problems
• Big computation, local optima and overfitting
• Limitations of learning prior knowledge
• Kernel function: Human Intervention
• Hierarchical feature Learning


NOOOOOO



Peeking into the Math

Gate Way to ML | ANN

Machine Learning Algorithms the Big picture

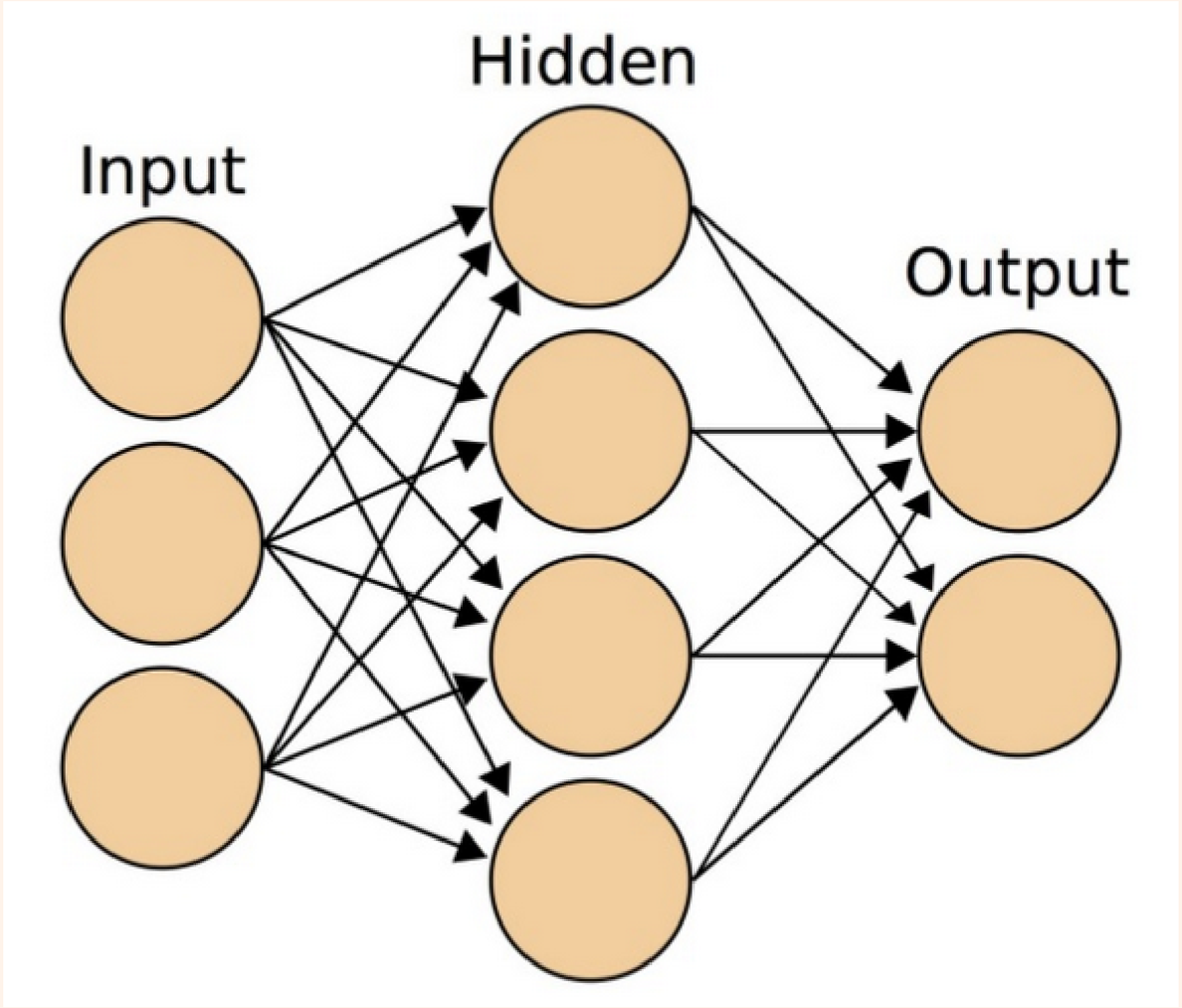
Model Representation

At a very simple level, neurons are basically computational units that take input (dendrites) as electrical input (called "spikes") that are channeled to outputs (axons).

In our model, our dendrites are like the input features x1…xn, and the output is the result of our hypothesis function.

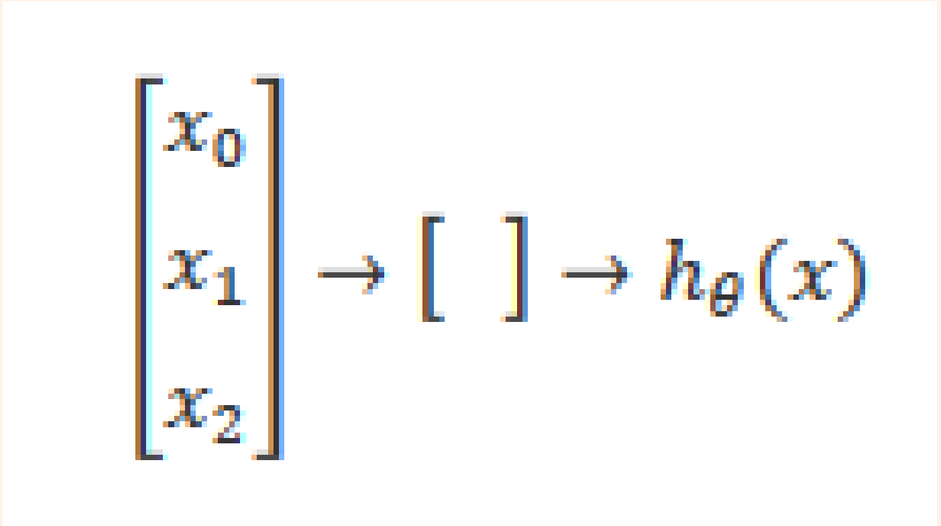
In this model our x0 input node is sometimes called the "bias unit." It is always equal to 1.

Structure of Artificial Neural Network



Model Representation

The"theta" parameters are called "weights" in the neural networks model.



Input nodes (layer 1) go into another node (layer 2), and are output as the hypothesis function.

The first layer is called the "input layer" and the final layer the "output layer," which gives the final value computed on the hypothesis.The intermediate layers of nodes between the input and output layers called the "hidden layer."

Model Representation

These intermediate or "hidden" layer nodes a02……an2 are "activation units."

$a_i^{(j)}$ = "activation" of unit i in layer j

$\Theta^{(j)}$ = matrix of weights controlling function mapping from layer j to layer $j+1$

If we had one hidden layer

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \rightarrow \begin{bmatrix} a_1^{(2)} \\ a_2^{(2)} \\ a_3^{(2)} \end{bmatrix} \rightarrow h_\theta(x)$$

Model Representation

The values for each of the "activation" nodes is obtained as follows:

$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$

$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

Where g(..) is the activation function

Activation Function

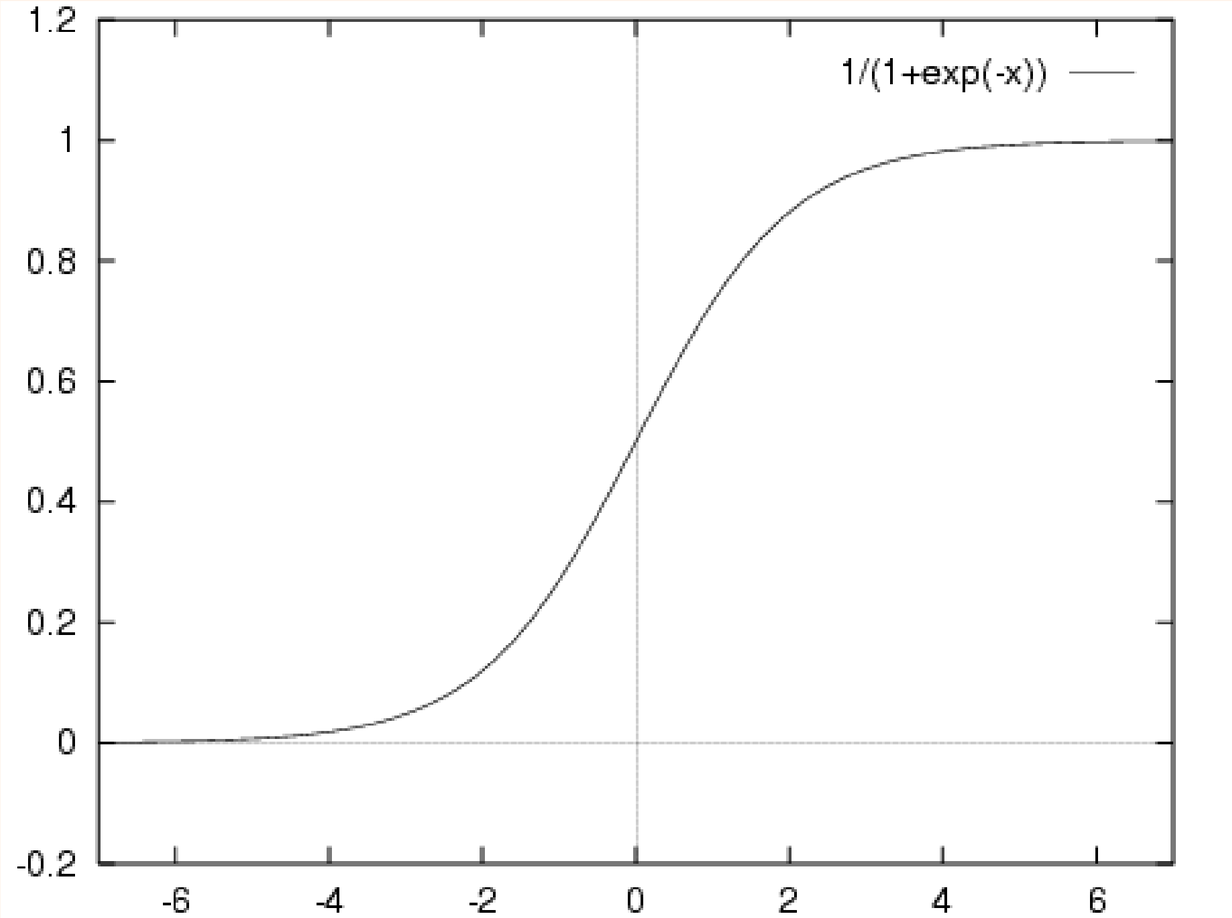
Activation functions are really important for a Artificial Neural Network to learn and make sense of something really complicated and Non-linear complex functional mappings between the inputs and response variable.They introduce non-linear properties to our Network.

Their main purpose is to convert a input signal of a node in a A-NN to an output signal. That output signal now is used as a input in the next layer in the stack.

Most popular types of Activation functions

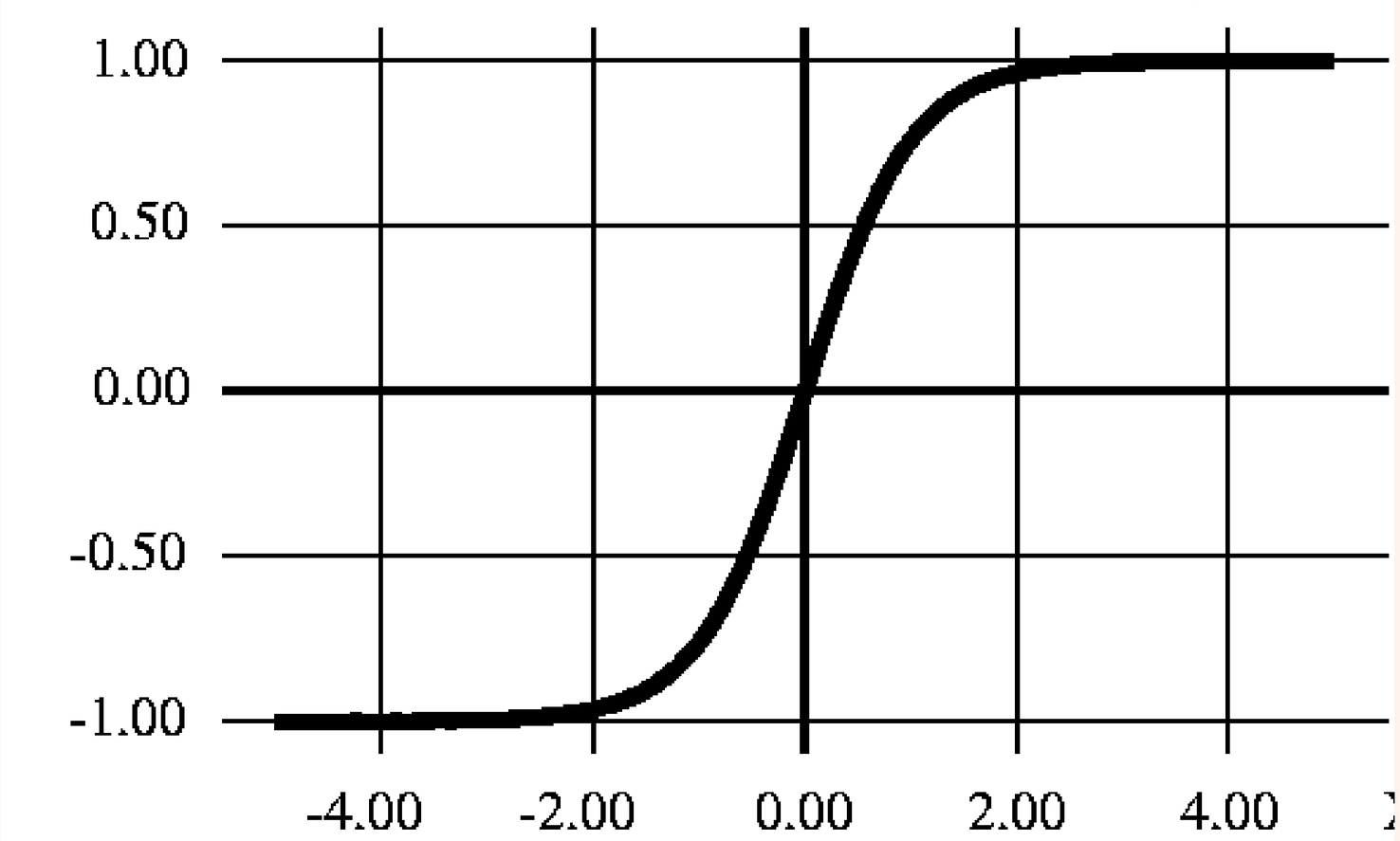
Sigmoid Activation function: It is a activation function of form f(x) = 1 / 1 + exp(-x) .

Its Range is between 0 and 1. It is a S—shaped curve.



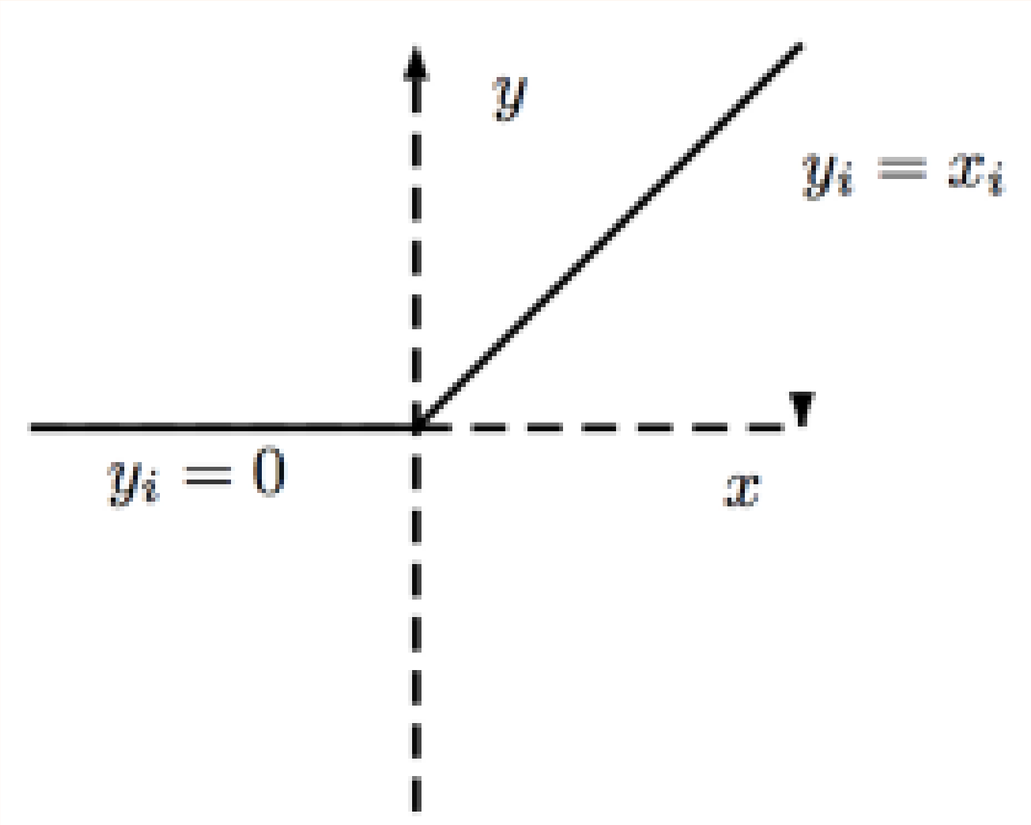
Most popular types of Activation functions

Hyperbolic Tangent function- Tanh : It's mathamatical formula is f(x) = 1— exp(-2x) / 1 + exp(-2x). Now it's output is zero centered because its range in between -1 to 1 i.e -1 < output < 1 .



Most popular types of Activation functions

ReLu- Rectified Linear units : It has become very popular in the past couple of years. It was recently proved that it had 6 times improvement in convergence from Tanh function. It's just R(x) = max(0,x) i.e if x < 0 , R(x) = 0 and if x >= 0 , R(x) = x.



Vectorzation of Model

A new variable zk(j) that encompasses the parameters inside our g function. In our previous example if we replaced the variable z for all the parameters we would get:

$$a_1^{(2)} = g(z_1^{(2)})$$
$$a_2^{(2)} = g(z_2^{(2)})$$
$$a_3^{(2)} = g(z_3^{(2)})$$

For layer j=2 and node k, the variable z will be:

$$z_k^{(2)} = \Theta_{k,0}^{(1)} x_0 + \Theta_{k,1}^{(1)} x_1 + \cdots + \Theta_{k,n}^{(1)} x_n$$

Vectorzation of Model

The vector representation of x and zj is:

$$x = \begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_n \end{bmatrix} \quad z^{(j)} = \begin{bmatrix} z_1^{(j)} \\ z_2^{(j)} \\ \cdots \\ z_n^{(j)} \end{bmatrix}$$

$$z^{(j)} = \Theta^{(j-1)} a^{(j-1)}$$

Setting x=a(1), we can rewrite the equation as:

Cost Function

$$cost(t) = y^{(t)} \log(h_\theta(x^{(t)})) + (1 - y^{(t)}) \log(1 - h_\theta(x^{(t)}))$$

Let

a) L= total number of layers in the network

b) sl = number of units (not counting bias unit) in layer l

c) K= number of output units/classes

$$J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}\left[y_k^{(i)} \log((h_\Theta(x^{(i)}))_k) + (1 - y_k^{(i)})\log(1 - (h_\Theta(x^{(i)}))_k)\right]$$

Cost Function With Regularization

$$J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}\left[y_k^{(i)}\log((h_\Theta(x^{(i)}))_k) + (1-y_k^{(i)})\log(1-(h_\Theta(x^{(i)}))_k)\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_{j,i}^{(l)})^2$$

The double sum simply adds up the costs calculated for each cell in the output layer.

The triple sum simply adds up the squares of all the individual Θs in the entire network.

The i in the triple sum does not refer to training example i.

Backpropagation Algorithm

"Backpropagation" is neural-network terminology for minimizing cost function

Our goal is to compute: min J(Θ)

In back propagation we're going to compute for every node:

$\delta_j^{(l)}$ = "error" of node j in layer l

Recall that $a_j^{(l)}$ is activation node j in layer l.

For the **last layer**, we can compute the vector of delta values with:

$$\delta^{(L)} = a^{(L)} - y$$

Where L is our total number of layers and a(L) is the vector of outputs of the activation units for the last layer. So our "error values" for the last layer are simply the differences of our actual results in the last layer and the correct outputs in y.

Back propagation Algorithm

- Set $a^{(1)} := x^{(t)}$
- Perform forward propagation to compute $a^{(l)}$ for l=2,3,...,L
- Using $y^{(t)}$, compute $\delta^{(L)} = a^{(L)} - y^{(t)}$
- Compute $\delta^{(L-1)}, \delta^{(L-2)}, \dots, \delta^{(2)}$ using $\delta^{(l)} = ((\Theta^{(l)})^T \delta^{(l+1)}) \, .* \, a^{(l)} \, .* \, (1 - a^{(l)})$
- $\Delta_{i,j}^{(l)} := \Delta_{i,j}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$ or with vectorization, $\Delta^{(l)} := \Delta^{(l)} + \delta^{(l+1)} (a^{(l)})^T$
- $D_{i,j}^{(l)} := \frac{1}{m} \left(\Delta_{i,j}^{(l)} + \lambda \Theta_{i,j}^{(l)} \right)$
- $D_{i,j}^{(l)} := \frac{1}{m} \Delta_{i,j}^{(l)}$ If j=0

The capital-delta matrix is used as an "accumulator" to add up our values as we go along and eventually compute our partial derivative.

The actual proof is quite involved, but, the $D_{i,j}^{(l)}$ terms are the partial derivatives and the results we are looking for:

$$D_{i,j}^{(l)} = \frac{\partial J(\Theta)}{\partial \Theta_{i,j}^{(l)}}.$$


Phew.

The Marketing term

The Intuitive Example

Intuitive
Artificial Neural Network

Input

w_1

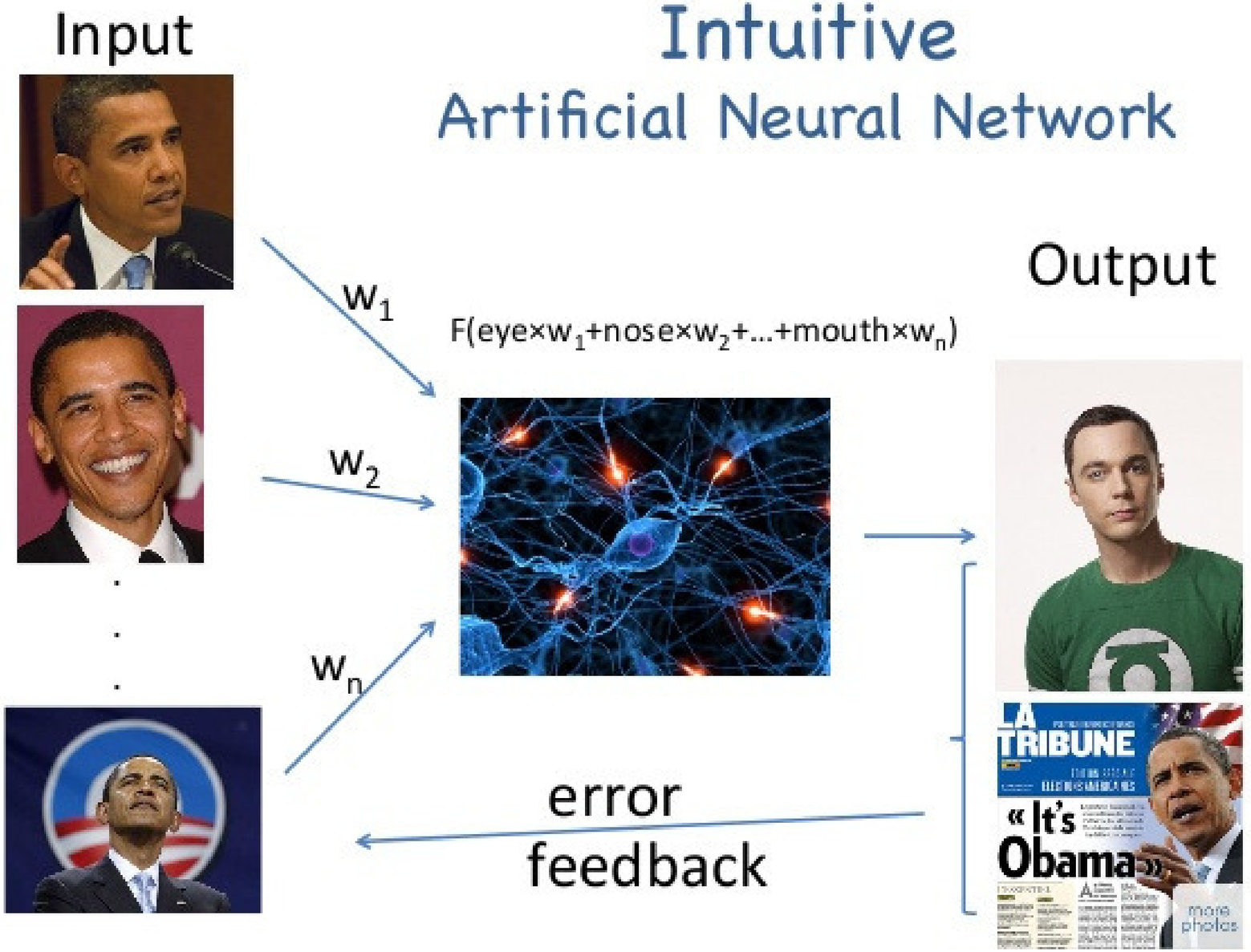
w_2

w_n

$F(eye \times w_1 + nose \times w_2 + \ldots + mouth \times w_n)$

Output

Intuitive
Artificial Neural Network



Intuitive
Artificial Neural Network

Input

w_1

w_2

w_n

$F(eye \times w_1 + nose \times w_2 + \ldots + mouth \times w_n)$

Output


THANK YOU FOR LISTENING
ANY QUESTIONS?

NEXT SECTION

Face detection system