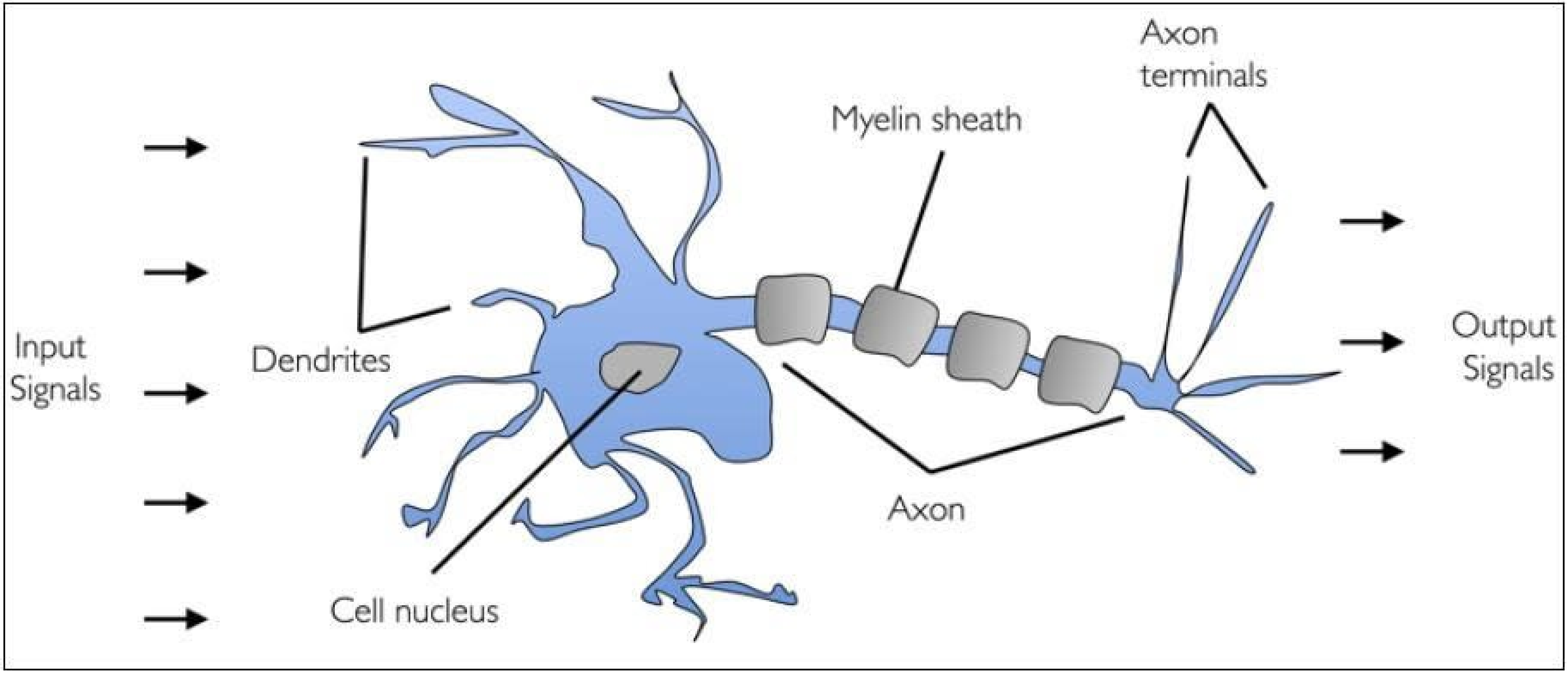
Perceptron

Tushar B. Kute,
http://tusharkute.com

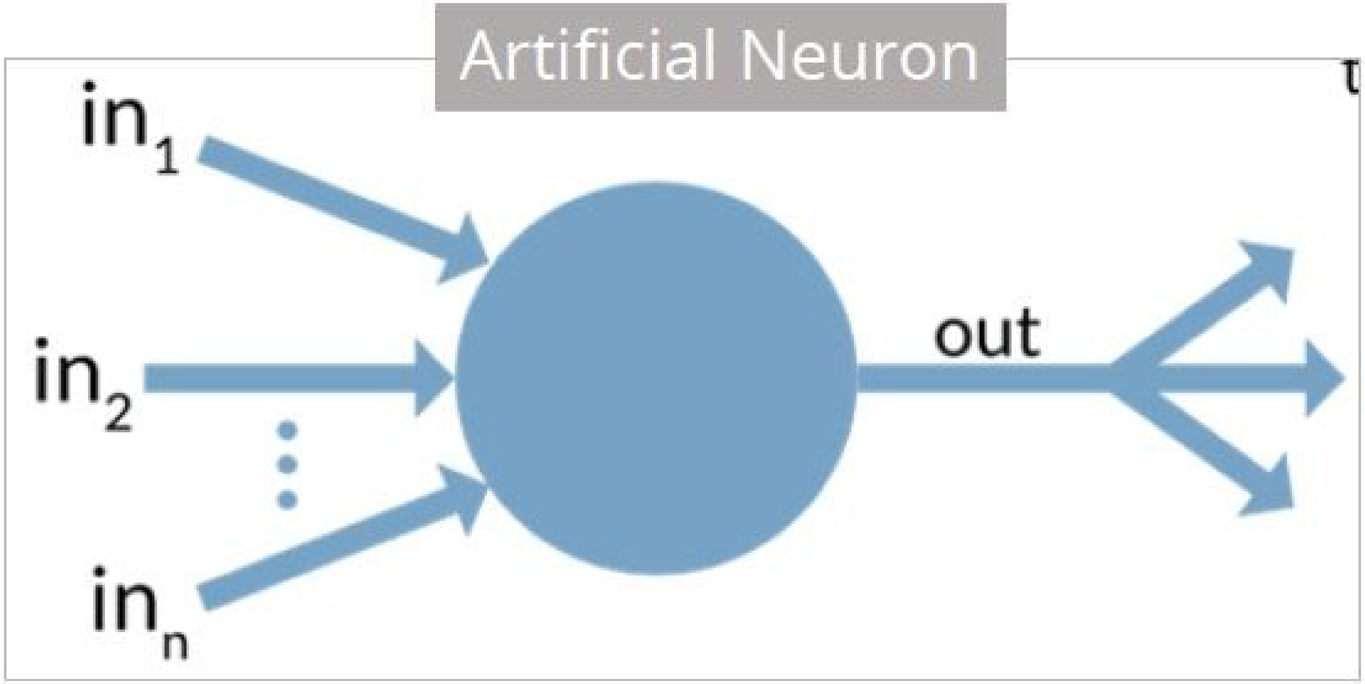
Biological Neuron



Artificial Neuron

- Researchers Warren McCullock and Walter Pitts published their first concept of simplified brain cell in 1943.
- This was called McCullock-Pitts (MCP) neuron. They described such a nerve cell as a simple logic gate with binary outputs.
- Multiple signals arrive at the dendrites and are then integrated into the cell body, and, if the accumulated signal exceeds a certain threshold, an output signal is generated that will be passed on by the axon.

Artificial Neuron



Biological vs. Artificial Neuron

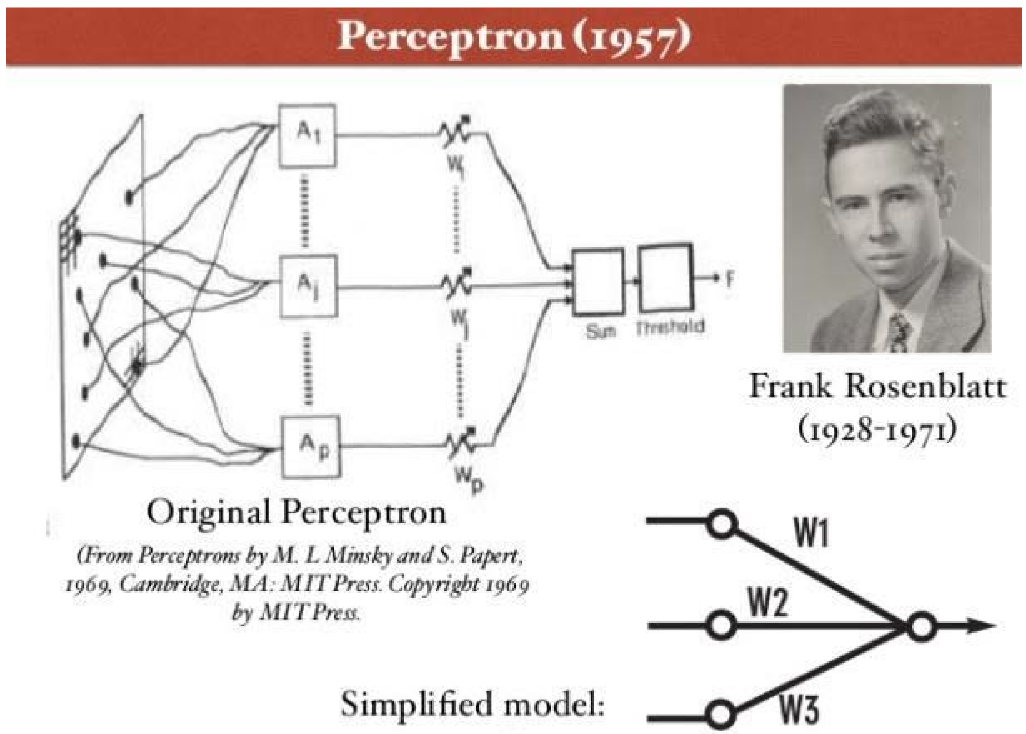
Biological Neuron	Artificial Neuron
Cell Nucleus (Soma)	Node
Dendrites	Input
Synapse	Weights or interconnections
Axon	Output

Artificial Neuron

- The artificial neuron has the following characteristics:
  - A neuron is a mathematical function modeled on the working of biological neurons
  - It is an elementary unit in an artificial neural network
  - One or more inputs are separately weighted
  - Inputs are summed and passed through a nonlinear function to produce output
  - Every neuron holds an internal state called activation signal
  - Each connection link carries information about the input signal
  - Every neuron is connected to another neuron via connection link

Perceptron

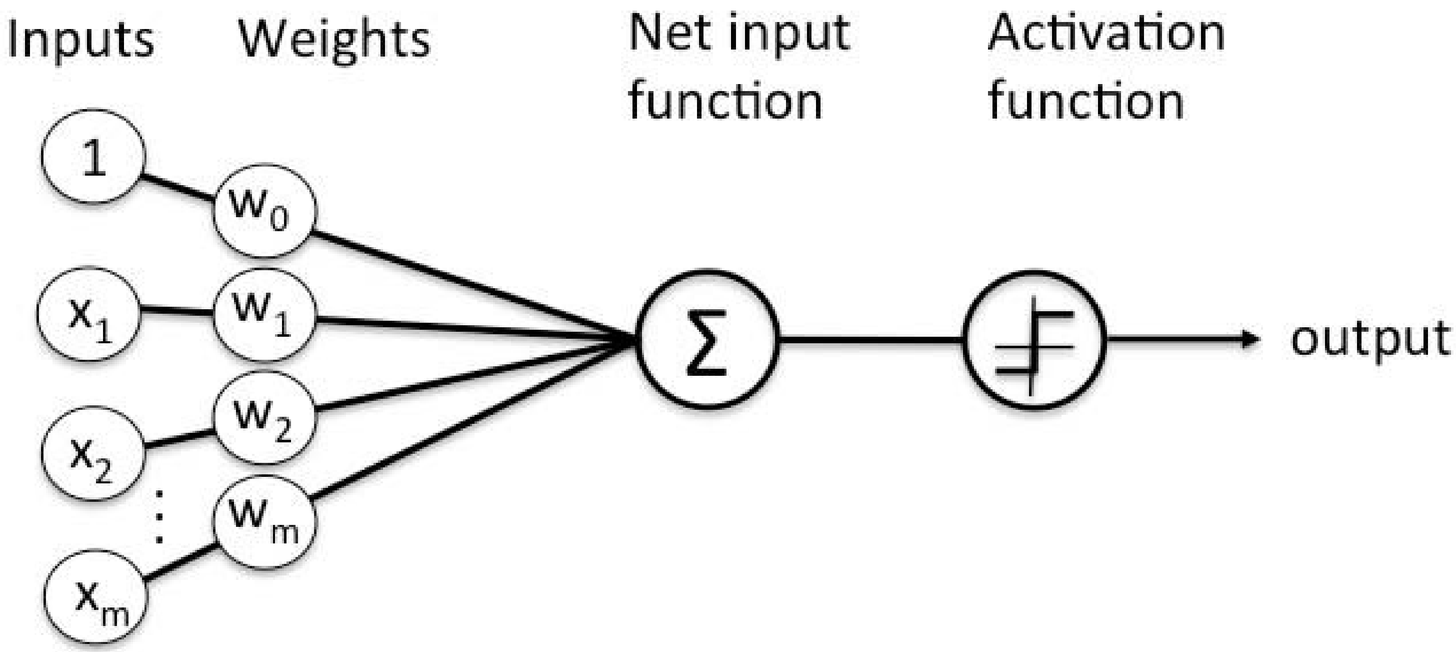
- A perceptron is a neural network unit (an artificial neuron) that does certain computations to detect features or business intelligence in the input data.



Perceptron

- Perceptron was introduced by Frank Rosenblatt in 1957.
- He proposed a Perceptron learning rule based on the original MCP neuron.
- A Perceptron is an algorithm for supervised learning of binary classifiers.
- This algorithm enables neurons to learn and processes elements in the training set one at a time.

Perceptron



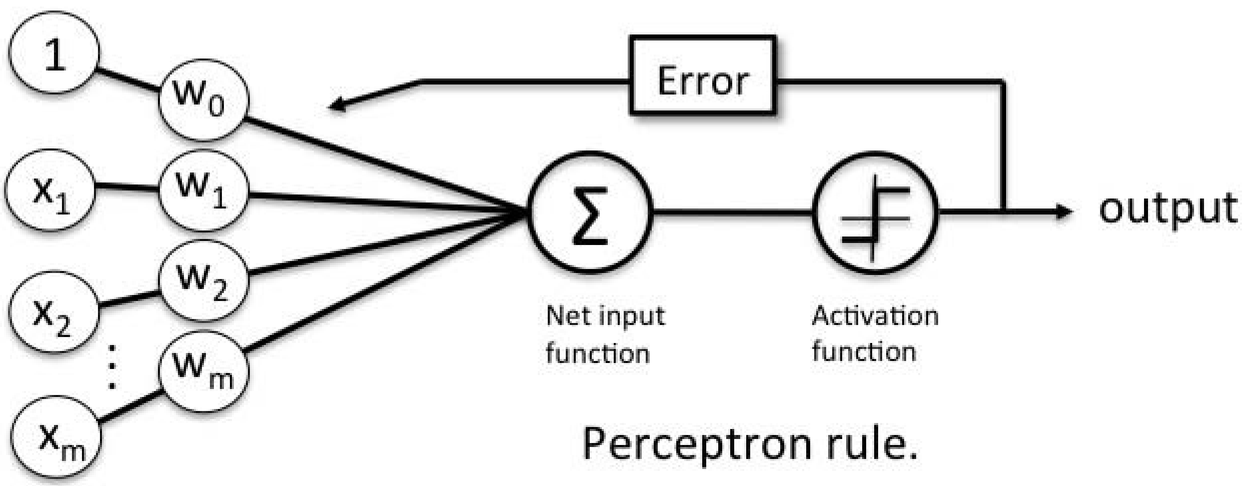
Inputs
Weights
Net input function
Activation function
1
w_0
x_1
w_1
x_2
w_2
x_m
w_m
Σ
output

Perceptron

- There are two types of Perceptrons: Single layer and Multilayer.
- Single layer Perceptrons can learn only linearly separable patterns.
- Multilayer Perceptrons or feedforward neural networks with two or more layers have the greater processing power.
- The Perceptron algorithm learns the weights for the input signals in order to draw a linear decision boundary.
- This enables you to distinguish between the two linearly separable classes +1 and -1.
- Note: Supervised Learning is a type of Machine Learning used to learn models from labeled training data. It enables output prediction for future or unseen data.

Perceptron Learning Rule

- Perceptron Learning Rule states that the algorithm would automatically learn the optimal weight coefficients.
- The input features are then multiplied with these weights to determine if a neuron fires or not.



Perceptron rule.

Perceptron function

- Perceptron is a function that maps its input “x,” which is multiplied with the learned weight coefficient; an output value ”f(x)”is generated.

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

- In the equation given above:

  “w” = vector of real-valued weights

  “b” = bias (an element that adjusts the boundary away from origin without any dependence on the input value)

  “x” = vector of input x values

Perceptron function

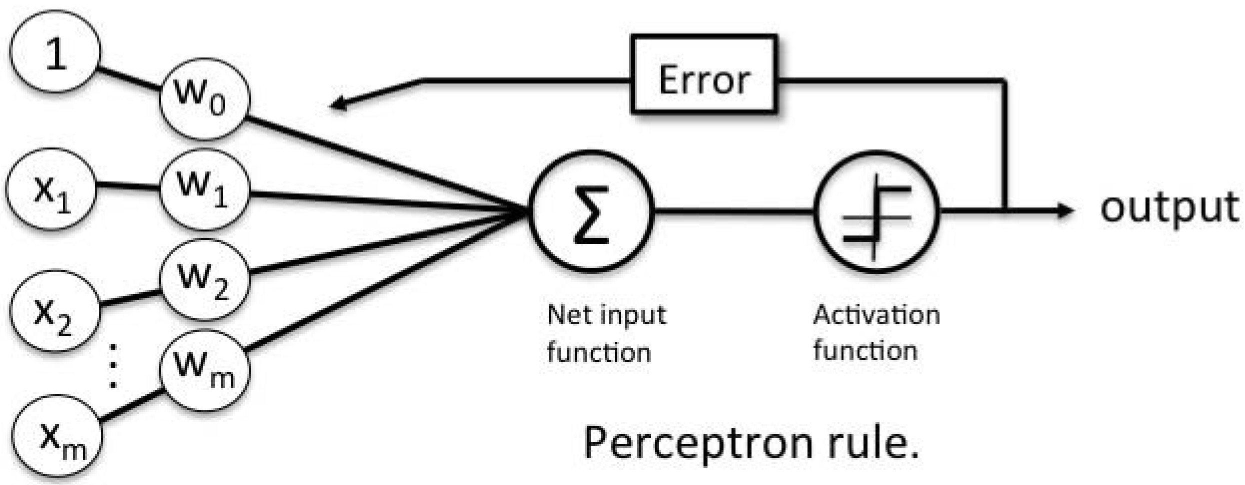
"m" = number of inputs to the Perceptron

$$\sum_{i=1}^{m} w_i x_i$$

- The output can be represented as "1" or "0." It can also be represented as "1" or "-1" depending on which activation function is used.

Inputs of Perceptron

- A Perceptron accepts inputs, moderates them with certain weight values, then applies the transformation function to output the final result. The above below shows a Perceptron with a Boolean output.



Perceptron rule.

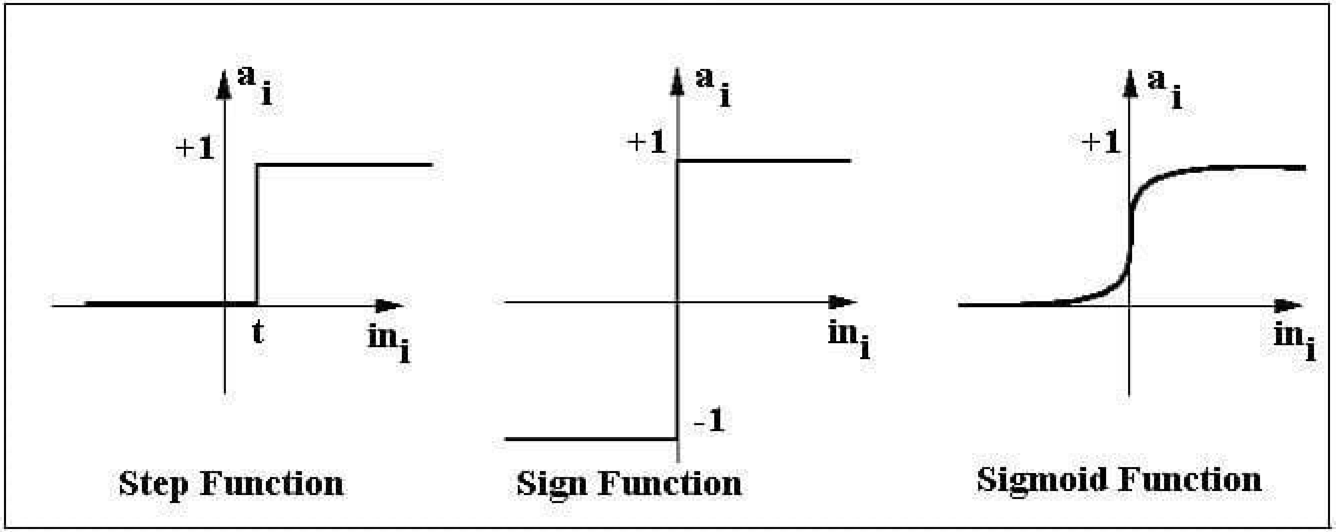
Inputs of Perceptron

- A Boolean output is based on inputs such as salaried, married, age, past credit profile, etc. It has only two values: Yes and No or True and False.
- The summation function "∑" multiplies all inputs of "x" by weights "w" and then adds them up as follows:

$$w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$

Activation function

- The activation function applies a step rule (convert the numerical output into +1 or -1) to check if the output of the weighting function is greater than zero or not.



Example

- For example:

  If $\sum wixi > 0$ => then final output "o" = 1 (issue bank loan)

  Else, final output "o" = -1 (deny bank loan)

- Step function gets triggered above a certain value of the neuron output; else it outputs zero. Sign Function outputs +1 or -1 depending on whether neuron output is greater than zero or not. Sigmoid is the S-curve and outputs a value between 0 and 1.

Output of Perceptron

- Perceptron with a Boolean output:

- Inputs: x1…xn

- Output: o(x1….xn)

$$o(x_1, \ldots, x_n) = \begin{cases} 1 \text{ if } w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n > 0 \\ -1 \text{ otherwise} \end{cases}$$

- Weights: wi=> contribution of input xi to the Perceptron output;

  w0=> bias or threshold

Output of Perceptron

- If ∑w.x > 0, output is +1, else -1. The neuron gets triggered only when weighted input reaches a certain threshold value.

$$o(\vec{x}) = sgn(\vec{w} \cdot \vec{x})$$

$$sgn(y) = \begin{cases} 1 & \text{if } y > 0 \\ -1 & \text{otherwise} \end{cases}$$

- An output of +1 specifies that the neuron is triggered. An output of -1 specifies that the neuron did not get triggered.
- "sgn" stands for sign function with output +1 or -1.

Error in Perceptron

- In the Perceptron Learning Rule, the predicted output is compared with the known output.
- If it does not match, the error is propagated backward to allow weight adjustment to happen.

Perceptron decision function

- A decision function φ(z) of Perceptron is defined to take a linear combination of x and w vectors.

$$\boldsymbol{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}$$

Perceptron: Decision Function

- The value z in the decision function is given by:

$$z = w_1 x_1 + \ldots + w_m x_m$$

- The decision function is +1 if z is greater than a threshold θ, and it is -1 otherwise.

$$\phi(z) = \begin{cases} 1 & if\ z \geq \theta \\ -1 & otherwise \end{cases}$$

- This is the Perceptron algorithm.

Bias Unit

- For simplicity, the threshold θ can be brought to the left and represented as w0x0, where w0= -θ and x0= 1.

$$z = w_0 x_0 + w_1 x_1 + \ldots + w_m x_m = \boldsymbol{w}^T \boldsymbol{x}$$

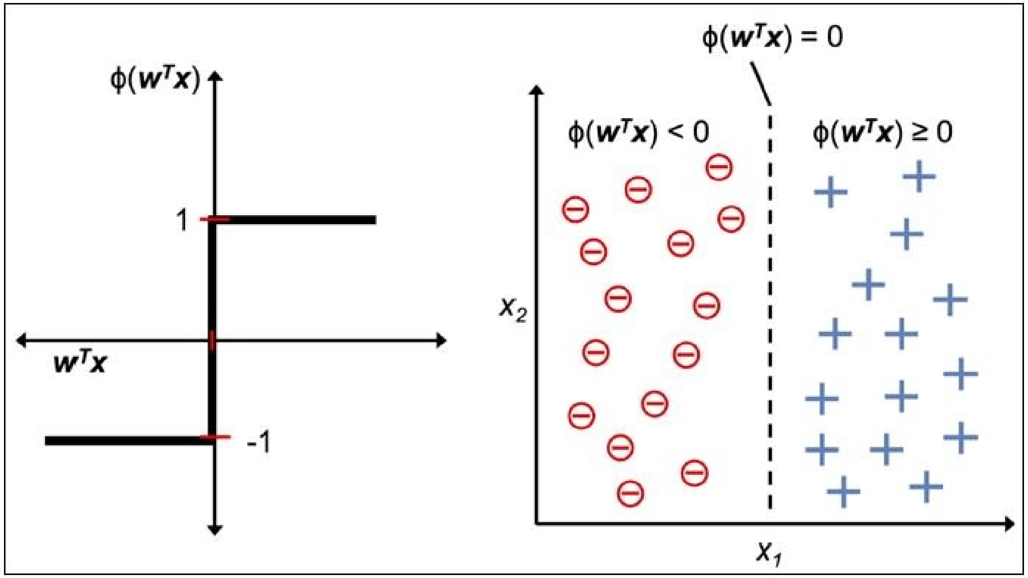
- The value w0 is called the bias unit.

- The decision function then becomes:

$$\phi(z) = \begin{cases} 1 & if\ z \geq 0 \\ -1 & otherwise \end{cases}$$

Output

- The figure shows how the decision function squashes wTx to either +1 or -1 and how it can be used to discriminate between two linearly separable classes.

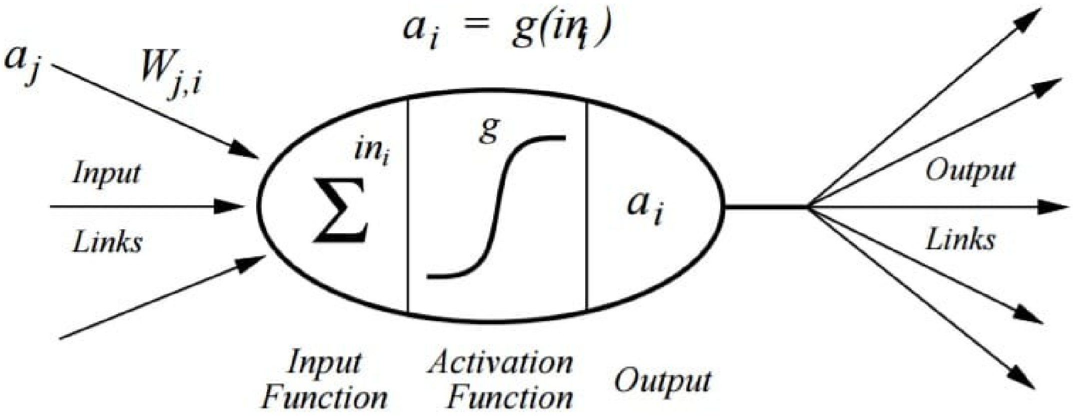


Perceptron at a Glance

- Perceptron has the following characteristics:
  - Perceptron is an algorithm for Supervised Learning of single layer binary linear classifier.
  - Optimal weight coefficients are automatically learned.
  - Weights are multiplied with the input features and decision is made if the neuron is fired or not.
  - Activation function applies a step rule to check if the output of the weighting function is greater than zero.
  - Linear decision boundary is drawn enabling the distinction between the two linearly separable classes +1 and -1.
  - If the sum of the input signals exceeds a certain threshold, it outputs a signal; otherwise, there is no output.

Sigmoid Activation Function

- The diagram given here shows a Perceptron with sigmoid activation function. Sigmoid is one of the most popular activation functions.



$$a_i = g(\sum_j W_{j,i} a_j)$$

Sigmoid Activation Function

- A Sigmoid Function is a mathematical function with a Sigmoid Curve ("S" Curve). It is a special case of the logistic function and is defined by the function given below:

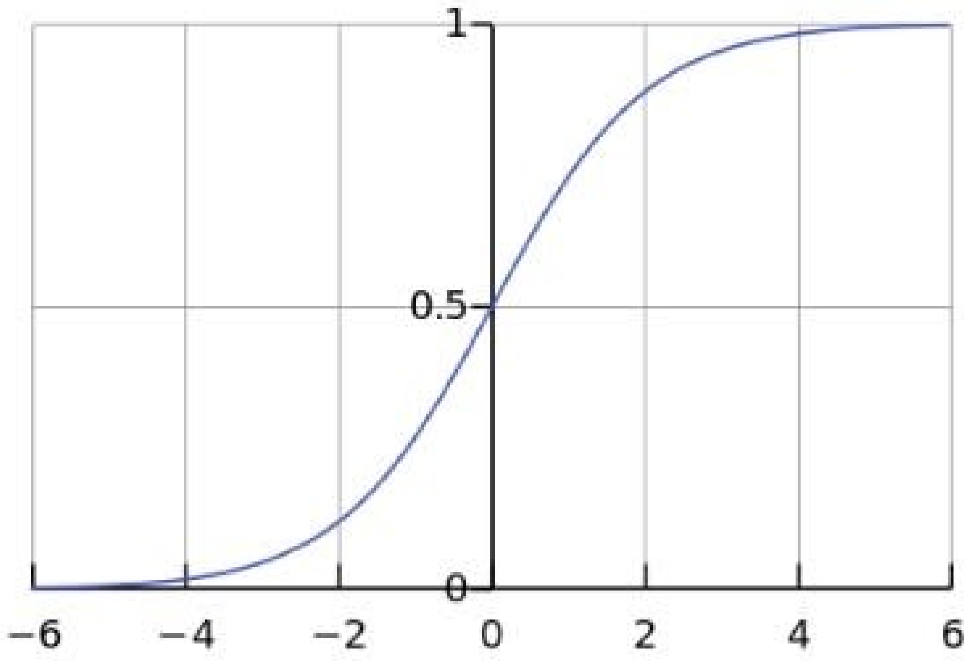
$$\phi_{logistic}(z) = \frac{1}{1+e^{-z}}$$

- Here, value of z is:

$$z = w_0x_0 + w_1x_1 + \cdots + w_mx_m = \sum_{i=0}^{m} w_ix_i = w^T x$$

Sigmoid Curve

- The curve of the Sigmoid function called “S Curve” is shown here.



Sigmoid Curve

- This is called a logistic sigmoid and leads to a probability of the value between 0 and 1.
- This is useful as an activation function when one is interested in probability mapping rather than precise values of input parameter t.
- The sigmoid output is close to zero for highly negative input.
- This can be a problem in neural network training and can lead to slow learning and the model getting trapped in local minima during training.
- Hence, hyperbolic tangent is more preferable as an activation function in hidden layers of a neural network.

Example:

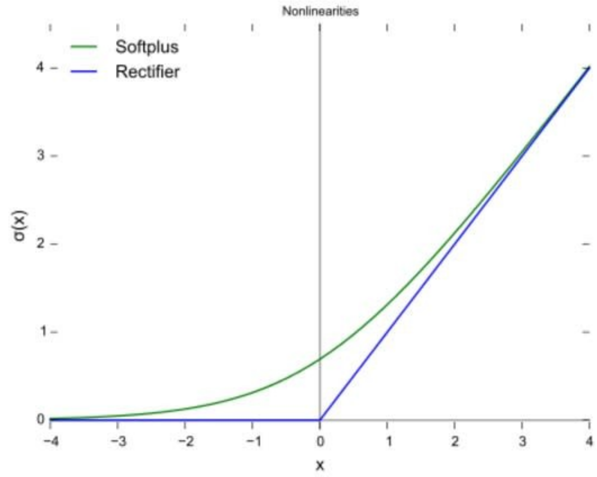
```
>>> import numpy as np

>>> X = np.array([1, 1.4, 2.5]) ## first value must be 1
>>> w = np.array([0.4, 0.3, 0.5])

>>> def net_input(X, w):
...     return np.dot(X, w)
...
>>> def logistic(z):
...     return 1.0 / (1.0 + np.exp(-z))
...
>>> def logistic_activation(X, w):
...     z = net_input(X, w)
...     return logistic(z)
...
>>> print('P(y=1|x) = %.3f' % logistic_activation(X, w))
P(y=1|x) = 0.888
```

Output

- The Perceptron output is 0.888, which indicates the probability of output y being a 1.
- If the sigmoid outputs a value greater than 0.5, the output is marked as TRUE.
- Since the output here is 0.888, the final output is marked as TRUE.
- In the next section, let us focus on the rectifier and softplus functions.

Relu and Softplus

- Apart from Sigmoid and Sign activation functions seen earlier, other common activation functions are ReLU and Softplus.
- They eliminate negative units as an output of max function will output 0 for all units 0 or less.



Relu and Softplus

- A rectifier or ReLU (Rectified Linear Unit) is a commonly used activation function.
- This function allows one to eliminate negative units in an ANN. This is the most popular activation function used in deep neural networks.
- A smooth approximation to the rectifier is the Softplus function:
- The derivative of Softplus is the logistic or sigmoid function:

Softmax Function

- Another very popular activation function is the Softmax function. The Softmax outputs probability of the result belonging to a certain set of classes. It is akin to a categorization logic at the end of a neural network.
- For example, it may be used at the end of a neural network that is trying to determine if the image of a moving object contains an animal, a car, or an airplane.
- In Mathematics, the Softmax or normalized exponential function is a generalization of the logistic function that squashes a K-dimensional vector of arbitrary real values to a K-dimensional vector of real values in the range (0, 1) that add up to 1.

Softmax Function

- In probability theory, the output of Softmax function represents a probability distribution over K different outcomes.
- In Softmax, the probability of a particular sample with net input z belonging to the ith class can be computed with a normalization term in the denominator, that is, the sum of all M linear functions:

$$p\left(y=i \middle| z\right)=\phi(z)=\frac{e^{z_i}}{\sum_{i=1}^{M} e^{z_j}}$$

- The Softmax function is used in ANNs and Naïve Bayes classifiers.

Softmax Function

- For example, if we take an input of [1,2,3,4,1,2,3], the Softmax of that is [0.024, 0.064, 0.175, 0.475, 0.024, 0.064, 0.175].
- The output has most of its weight if the original input is '4'
- This function is normally used for:
  - Highlighting the largest values
  - Suppressing values that are significantly below the maximum value.

Softmax Function

```
>>> def softmax(z):
...     return np.exp(z) / np.sum(np.exp(z))
...
>>> y_probas = softmax(Z)
>>> print('Probabilities:\n', y_probas)
Probabilities:
 [ 0.44668973  0.16107406  0.39223621]

>>> np.sum(y_probas)
1.0
```

This code implements the softmax formula and prints the probability of belonging to one of the three classes. The sum of probabilities across all classes is 1.

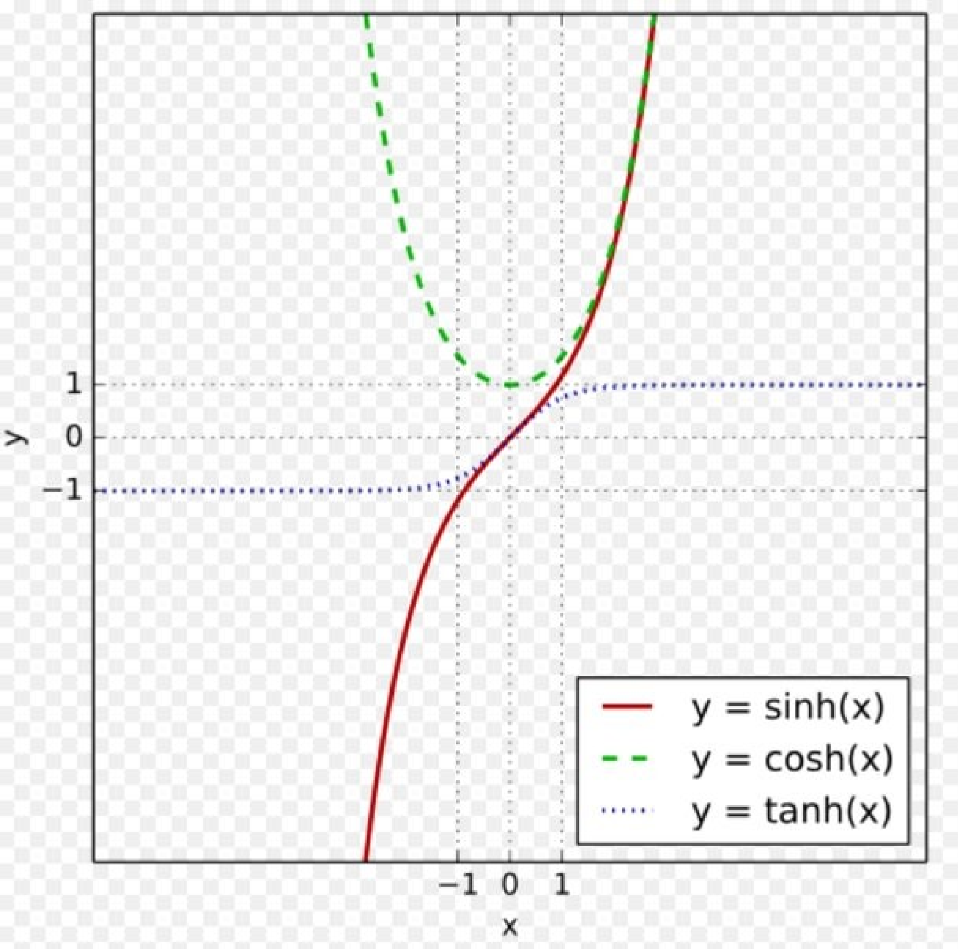
Hyperbolic Tangent

- Hyperbolic or tanh function is often used in neural networks as an activation function. It provides output between -1 and +1. This is an extension of logistic sigmoid; the difference is that output stretches between -1 and +1 here.

$$\phi_{tanh}(z) = 2 \times \phi_{logistic}(2z) - 1 = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

- The advantage of the hyperbolic tangent over the logistic function is that it has a broader output spectrum and ranges in the open interval (-1, 1), which can improve the convergence of the backpropagation algorithm.

Hyperbolic Activation Functions

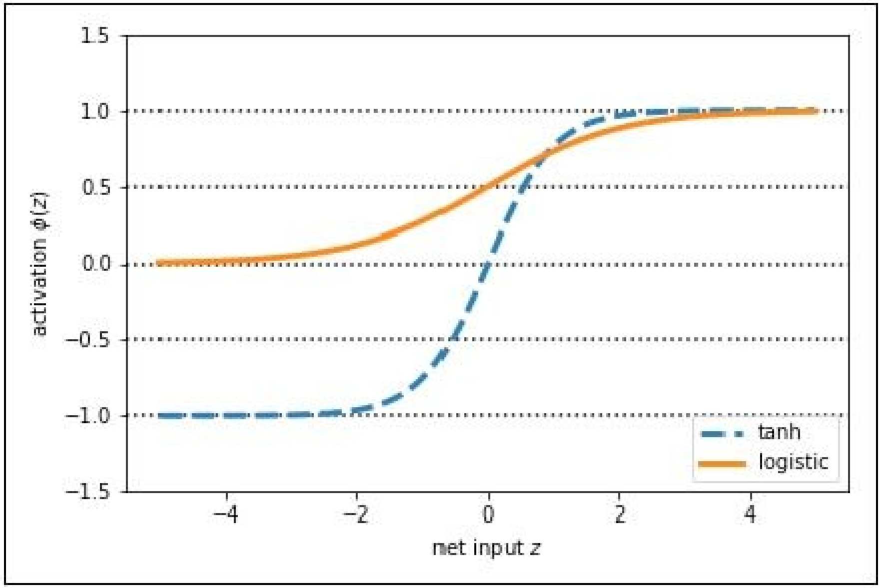


Hyperbolic Tangent

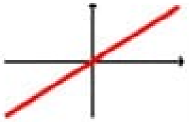
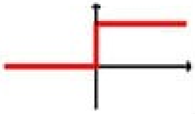
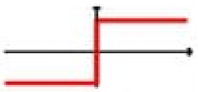
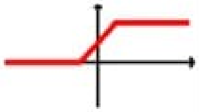
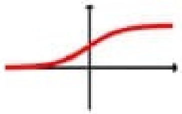
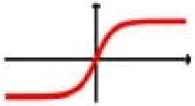
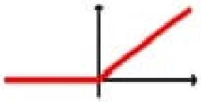
```
>>> def tanh(z):
...     e_p = np.exp(z)
...     e_m = np.exp(-z)
...     return (e_p - e_m) / (e_p + e_m)

>>> z = np.arange(-5, 5, 0.005)
>>> log_act = logistic(z)
>>> tanh_act = tanh(z)
```

Hyperbolic Tangent

- This code implements the tanh formula. Then it calls both logistic and tanh functions on the z value.
- The tanh function has two times larger output space than the logistic function.



Summary

Activation Function	Equation	Example	1D Graph
Linear	$\phi(z) = z$	Adaline, linear regression	
Unit Step (Heaviside Function)	$\phi(z) = \begin{cases} 0 & z < 0 \\ 0.5 & z = 0 \\ 1 & z > 0 \end{cases}$	Perceptron variant	
Sign (signum)	$\phi(z)= \begin{cases} -1 & z < 0 \\ 0 & z = 0 \\ 1 & z > 0 \end{cases}$	Perceptron variant	
Piece-wise Linear	$\phi(z)= \begin{cases} 0 & z \le -\frac{1}{2} \\ z + \frac{1}{2} & -\frac{1}{2} \le z \le \frac{1}{2} \\ 1 & z \ge \frac{1}{2} \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z)= \dfrac{1}{1 + e^{-z}}$	Logistic regression, Multilayer NN	
Hyperbolic Tangent (tanh)	$\phi(z)= \dfrac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multilayer NN, RNNs	
ReLU	$\phi(z)= \begin{cases} 0 & z < 0 \\ z & z > 0 \end{cases}$	Multilayer NN, CNNs	

Summary

- An artificial neuron is a mathematical function conceived as a model of biological neurons, that is, a neural network.
- A Perceptron is a neural network unit that does certain computations to detect features or business intelligence in the input data. It is a function that maps its input “x,” which is multiplied by the learned weight coefficient, and generates an output value ”f(x).
- ”Perceptron Learning Rule states that the algorithm would automatically learn the optimal weight coefficients.
- Single layer Perceptrons can learn only linearly separable patterns.
- Multilayer Perceptron or feedforward neural network with two or more layers have the greater processing power and can process non-linear patterns as well.
- Perceptrons can implement Logic Gates like AND, OR, or XOR.

References

- https://www.simplilearn.com

Thank you

@mitu_skillologies

/mITuSkillologies

@mitu_group

/company/mitu-skillologies

MITUSkillologies

Web Resources
https://mitu.co.in
http://tusharkute.com

contact@mitu.co.in

tushar@tusharkute.com