Introduction to Neural Networks

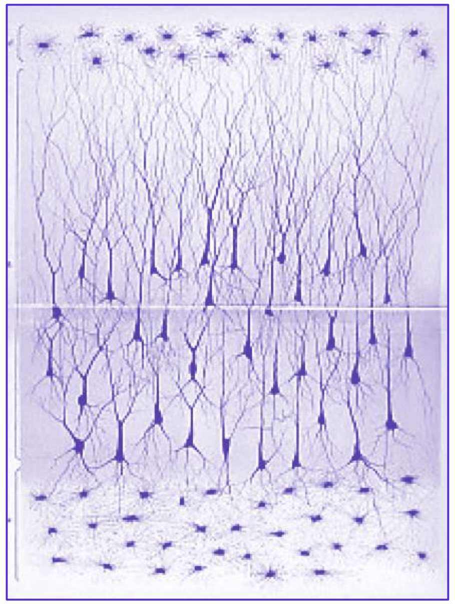
Dr. Tushar Sandhan

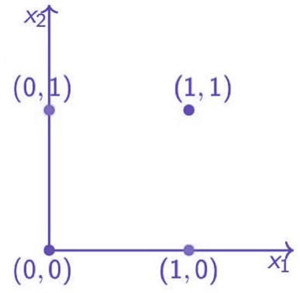
Contents

Historical Background of Neural Network

- Neurons
  - Camillo Golgi, 1872 (examined nervous tissue) – single unit theory
  - Cajal – neurons multicell joint system theory
  - Nobel prize 1906 (Golgi & Cajal)
  - HW Gottfried, W Hartz – coined the term

- Perceptron
  - “it may eventually be able to learn, make decisions and translate languages”
    - Frank Rosenblatt, 1958
  - MLP: Multilayer perceptron (Ivakhnenko, 1968)
  - Limitations of (less layer) MLP (Minsky & Papert, 1969)
  - Backpropogation (Werbos, Rumelhart, 1982)
  - Gradient descent (Cauchy, 1847)
  - Universal approximation theorem (1989)


Neurons in staining

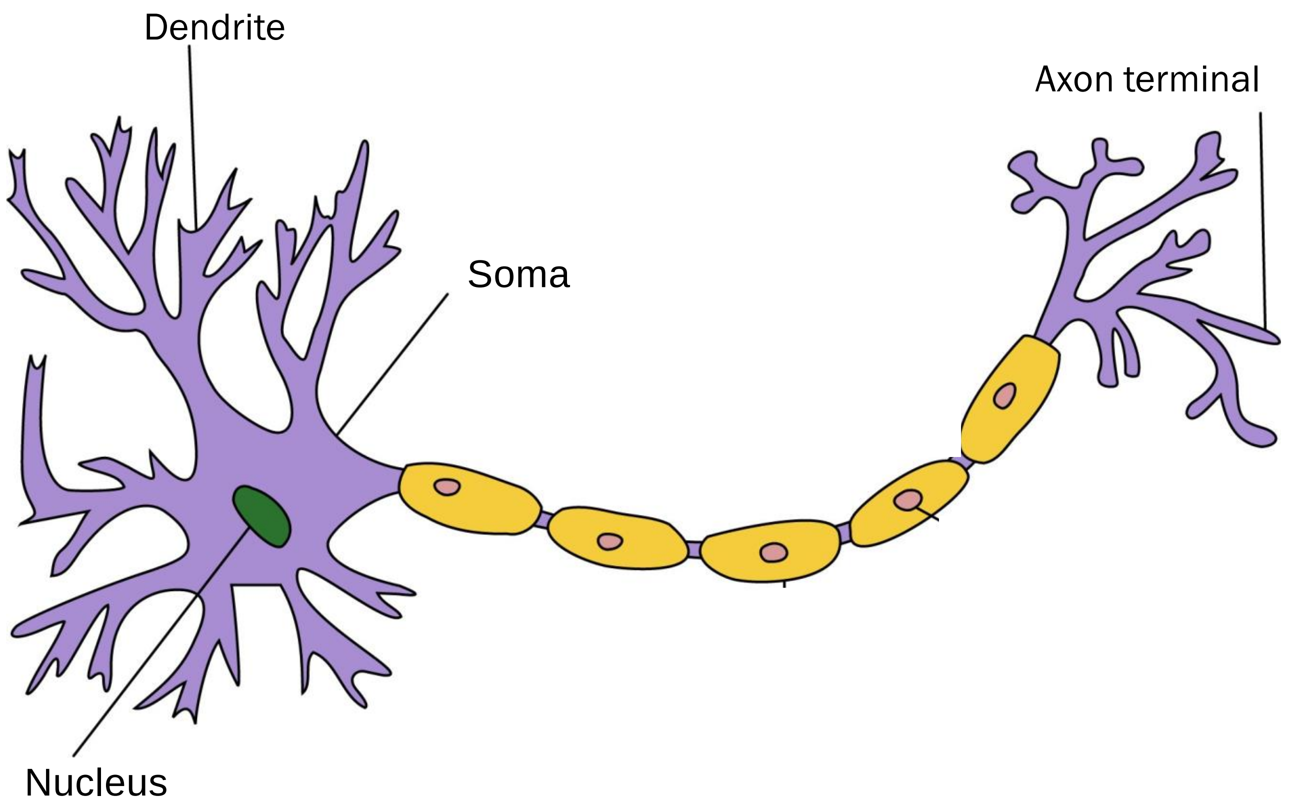


Nervous Systems

- THuman brain contains ~ 10^{11} neurons.
- Each neuron is connected ~ 10^{4} others.
- Some scientists compared the brain with a “complex, nonlinear, parallel computer”.
- The largest modern neural networks achieve the complexity comparable to a nervous system of a fly.

Neuron in the Brain

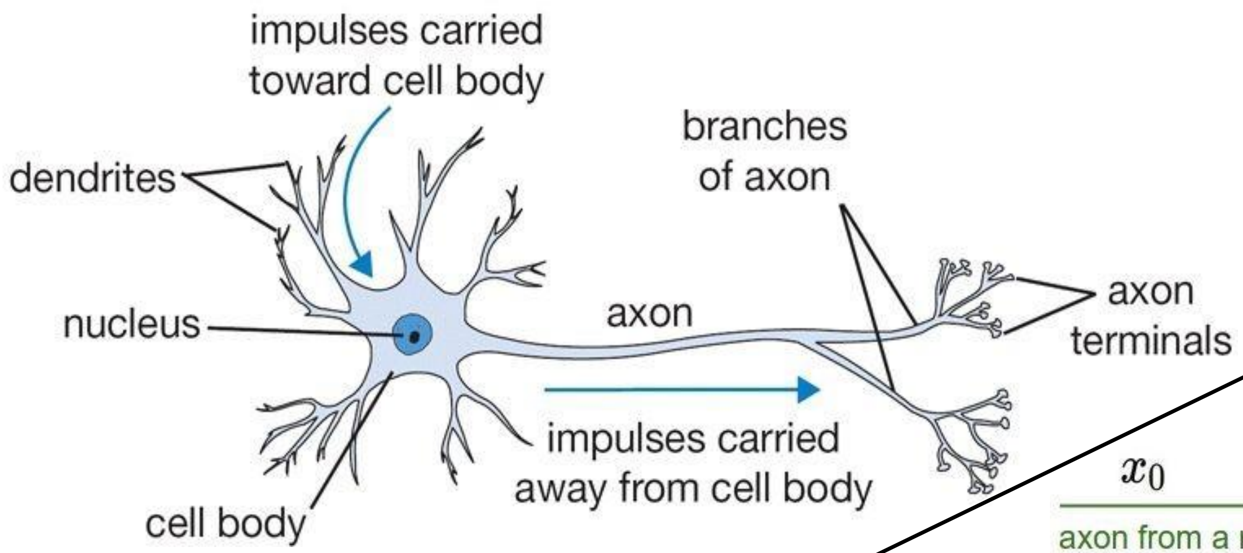
- The human brain is made up of about 100 billion neurons

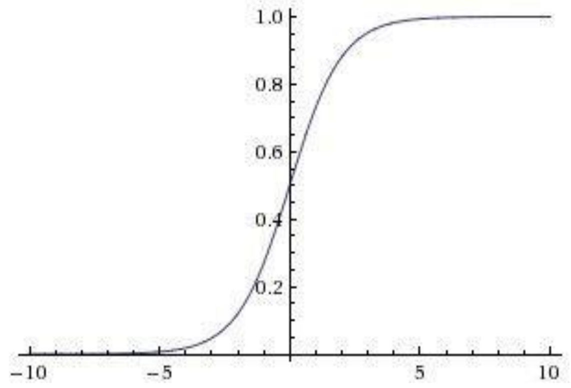


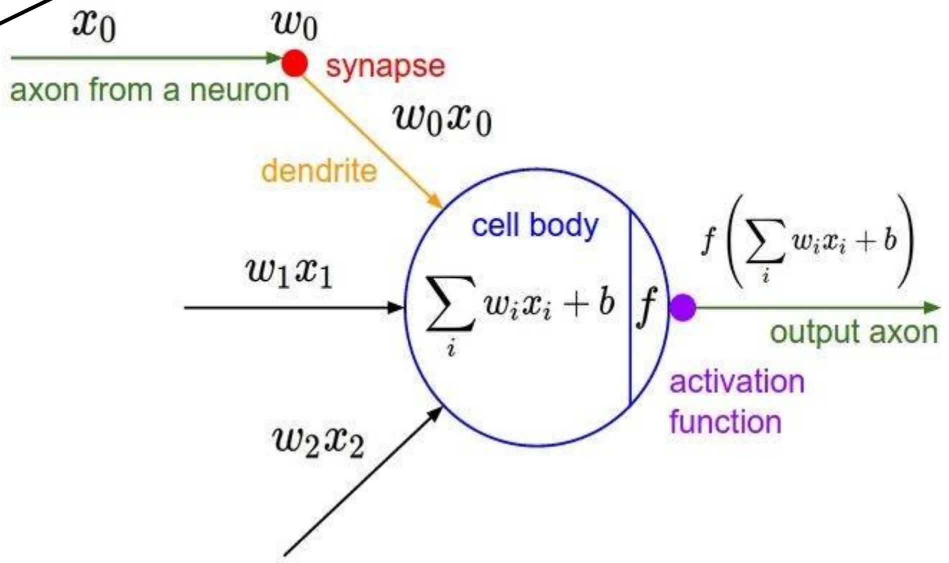
- Neurons receive electric signals at the dendrites and send them to the axon

sandhan@iitk.ac.in

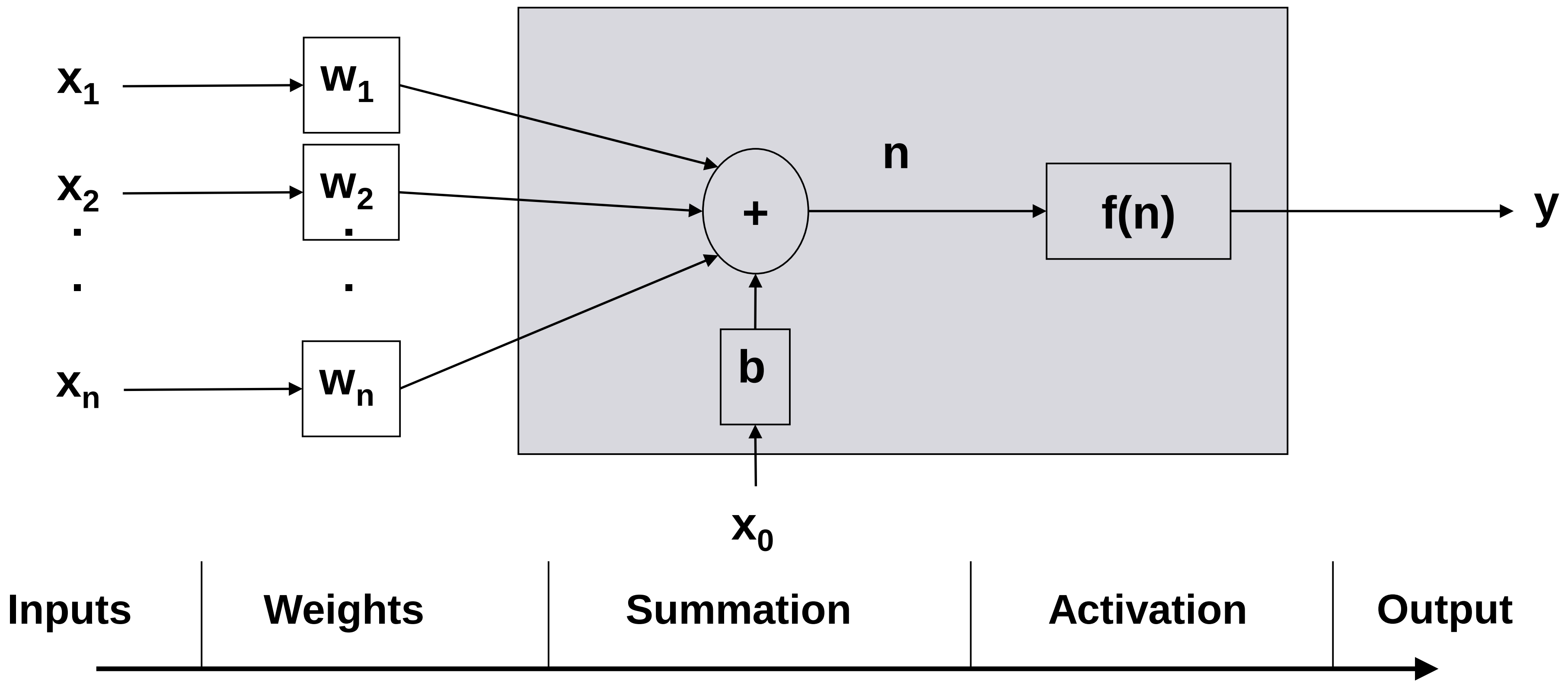
Biological Neuron Vs Artificial Neural Network


impulses carried toward cell body
dendrites
branches of axon
axon
axon terminals
nucleus
impulses carried away from cell body
cell body



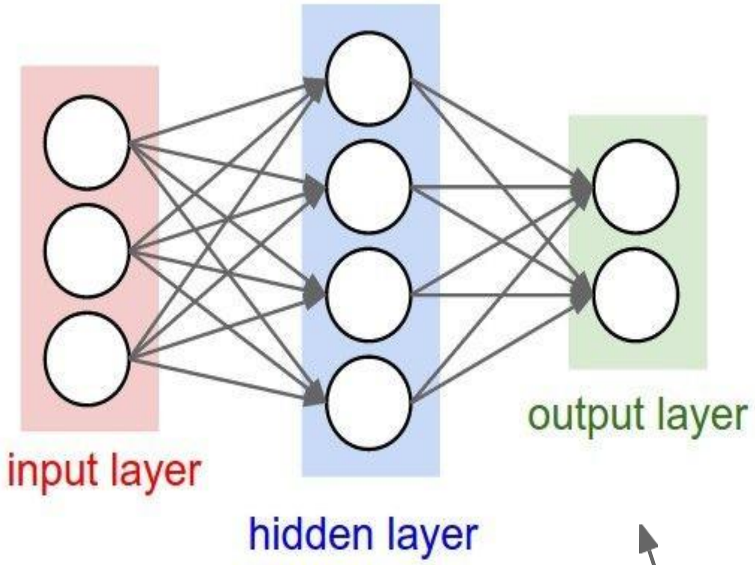


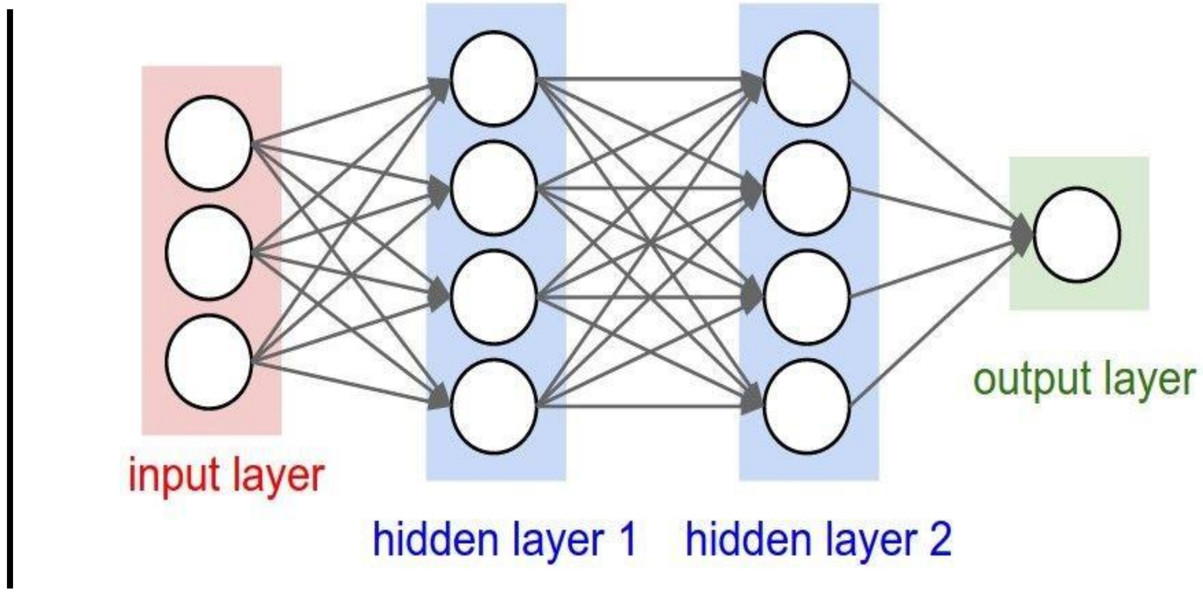
Artificial Neural Network (ANN)



Neural Networks: Architectures

Dropout (probability): dropout(0.2)



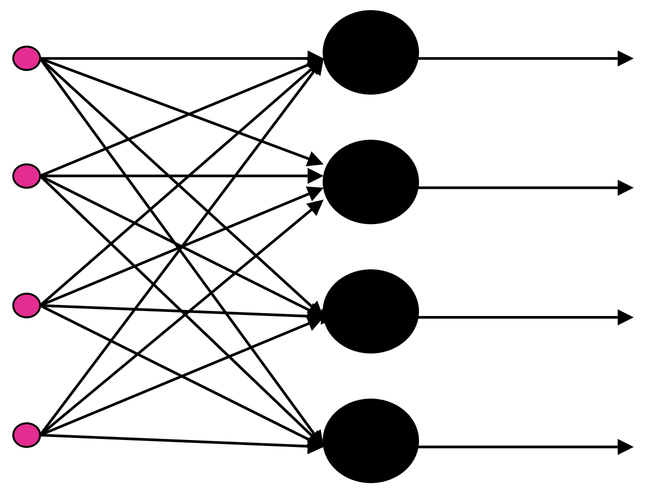


"2-layer Neural Net", or
"1-hidden-layer Neural Net"

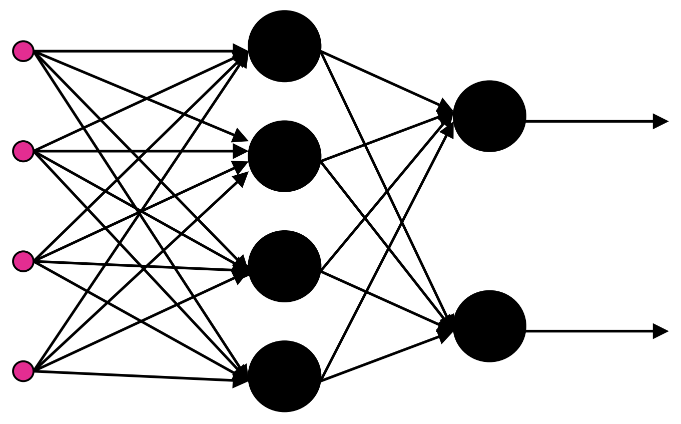
"Fully-connected" layers

"3-layer Neural Net", or
"2-hidden-layer Neural Net"

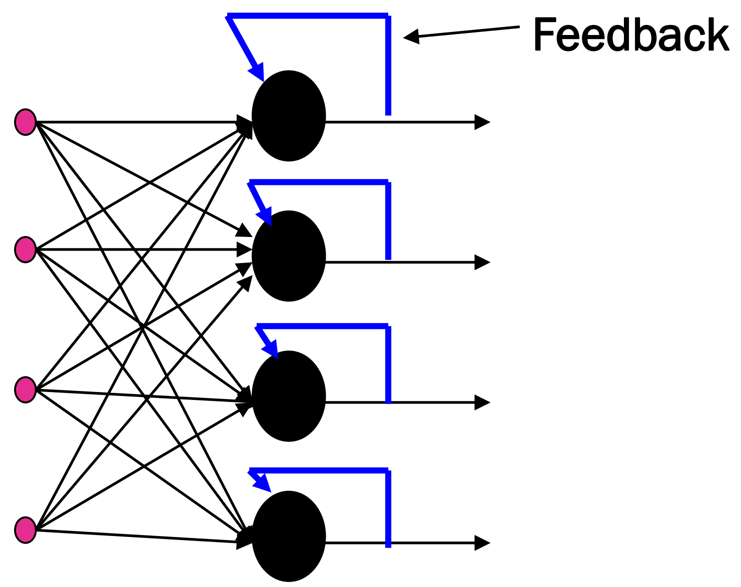
Types of networks



Multiple Inputs and Single Layer



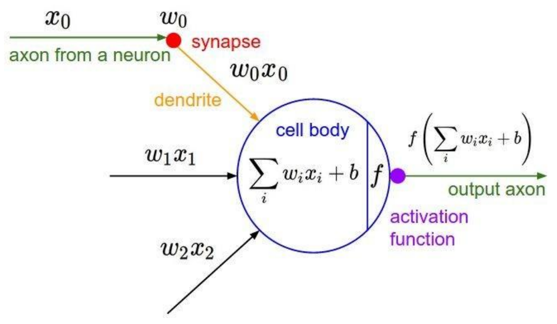
Multiple Inputs and multiple layers



Recurrent Networks

Why Activation Function

- **An Activation Function** decides whether a neuron should be activated or not. This means that it will decide whether the neuron's input to the network is important or not in the process of prediction using simple mathematical operations.
  - The role of the Activation Function is to derive output from a set of input values fed to a node
  - the purpose of an activation function is to add non-linearity to the neural network.



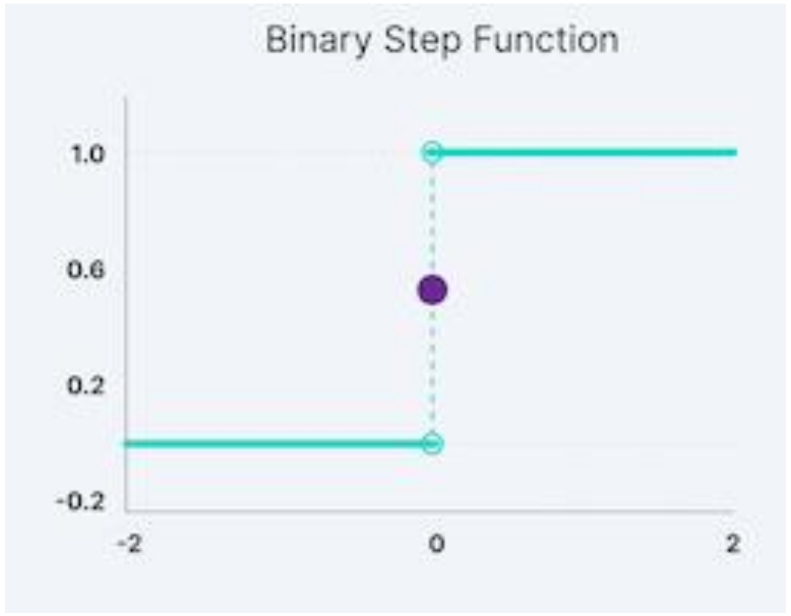
- **Let's suppose we have a neural network working without the activation functions.**
  - In that case, every neuron will only be performing a linear transformation on the inputs using the weights and biases. It's because it doesn't matter how many hidden layers we attach in the neural network; all layers will behave in the same way because the composition of two linear functions is a linear function itself.

Activation functions of NN

- **Binary Step Function:** Binary step function depends on a threshold value that decides whether a neuron should be activated or not.

$$f(x) = \begin{cases} 0 & for\ x < 0 \\ 1 & for\ x \geqslant 0 \end{cases}$$

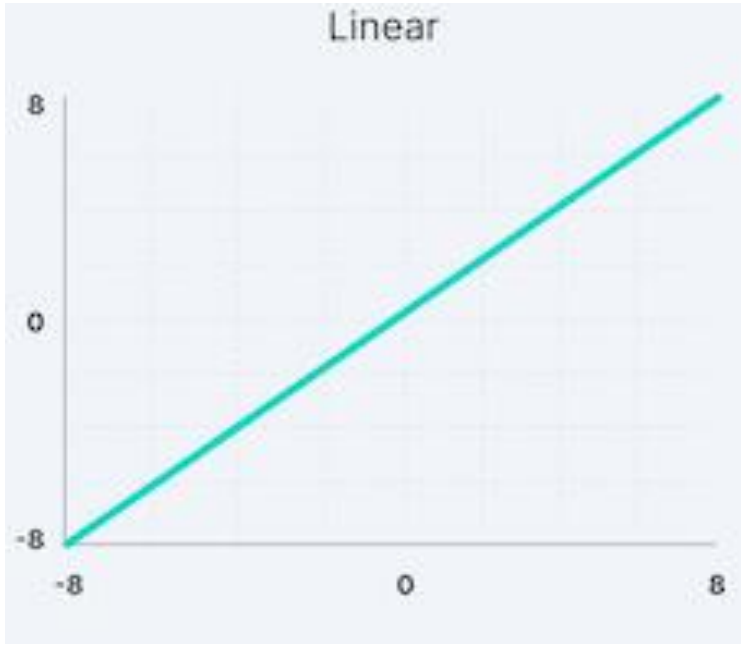
- **Limitations of binary step function:**
  - Sudden jump in the gradient at zero region
  - The gradient of the step function is zero, which causes a hindrance in the backpropagation process.



Activation functions of NN

- **Linear Activation Function:** The linear activation function, also known as "no activation," or "identity function" (multiplied x1.0), is where the activation is proportional to the input.
  - The function doesn't do anything to the weighted sum of the input, it simply spits out the value it was given.

$$f(x) = x$$



- **Limitations:**
  - It’s not possible to use backpropagation as the derivative of the function is a constant and has no relation to the input x.
  - All layers of the neural network will collapse into one if a linear activation function is used. No matter the number of layers in the neural network, the last layer will still be a linear function of the first layer. So, essentially, a linear activation function turns the neural network into just one layer.

Non-Linear Activation Functions

- The linear activation function does not allow the model to create complex mappings between the network's inputs and outputs.

- Non-linear activation functions solve the following limitations of linear activation functions:

  - They allow backpropagation because now the derivative function would be related to the input, and it's possible to go back and understand which weights in the input neurons can provide a better prediction.

  - They allow the stacking of multiple layers of neurons as the output would now be a non-linear combination of input passed through multiple layers. Any output can be represented as a functional computation in a neural network.

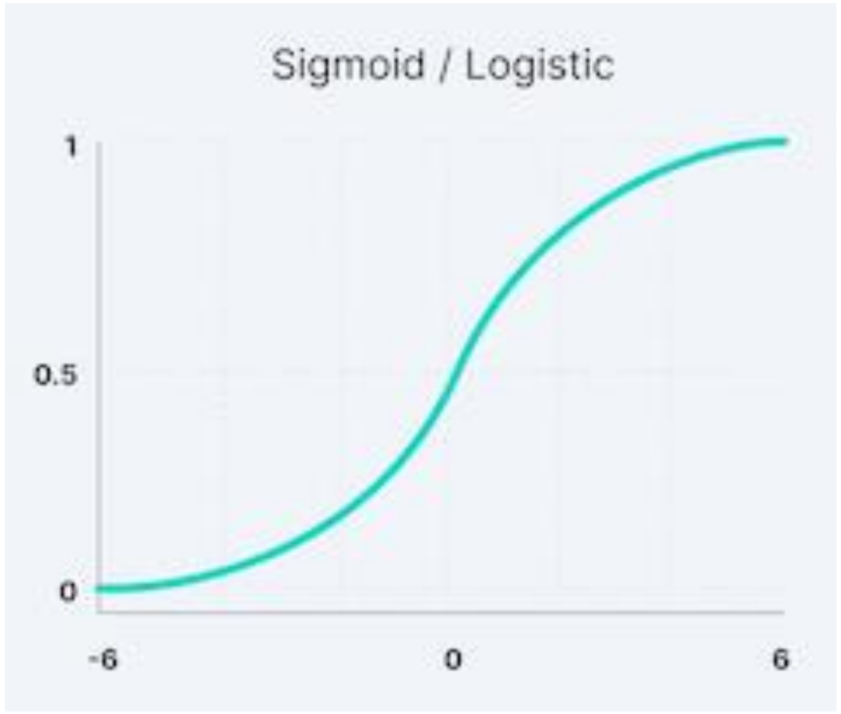
Non-Linear Activation Functions

- **Sigmoid / Logistic Activation Function:** This function takes any real value as input and outputs values in the range of 0 to 1.

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

- The larger the input (more positive), the closer the output value will be to 1.0, whereas the smaller the input (more negative), the closer the output will be to 0.0, as shown below.

- Sigmoid / logistic activation function is one of the most widely used functions

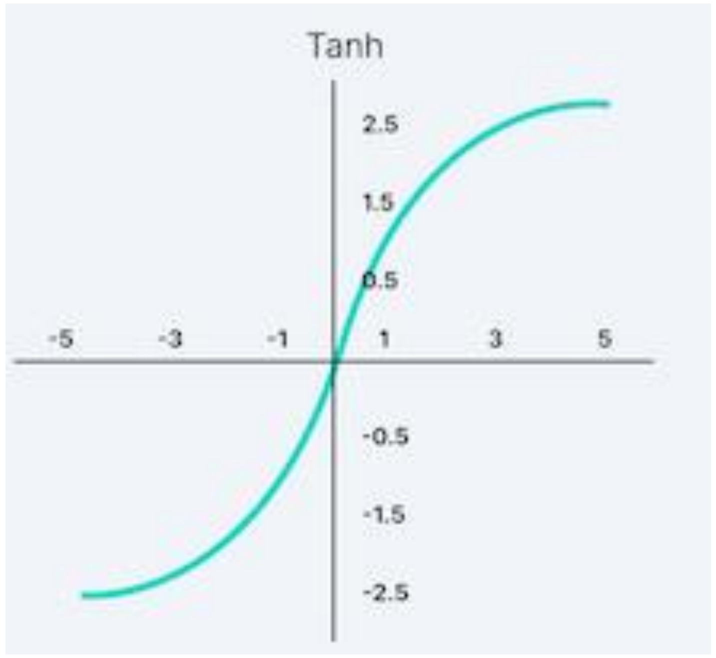


Non-Linear Activation Functions

- **Tanh Function (Hyperbolic Tangent):** Tanh function is very similar to the sigmoid/logistic activation function, and even has the same S-shape with the difference in output range of -1 to 1.

- In Tanh, the larger the input (more positive), the closer the output value will be to 1.0, whereas the smaller the input (more negative), the closer the output will be to -1.0.

$$f(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$$



Non-Linear Activation Functions

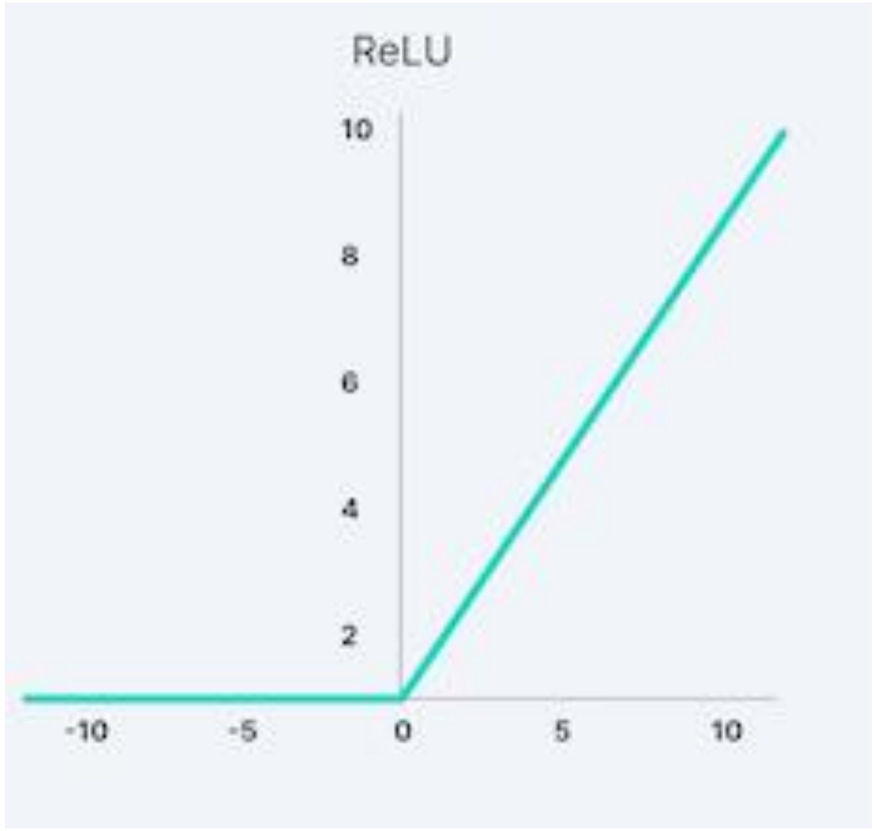
- **ReLU Function:** ReLU stands for Rectified Linear Unit.

$$f(x) = max\ (0, x)$$

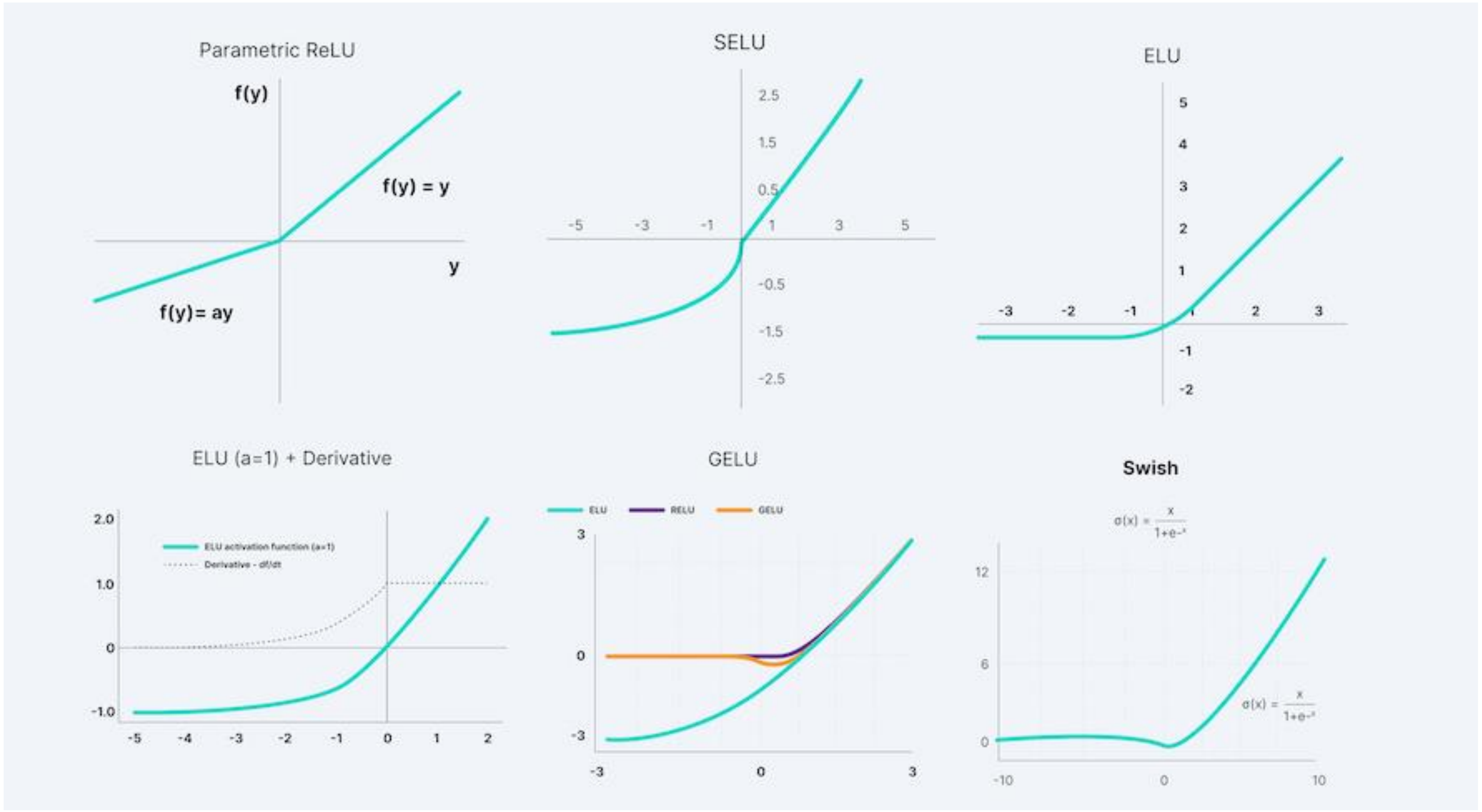
- Although it gives an impression of a linear function, ReLU has a derivative function and allows for backpropagation while simultaneously making it computationally efficient.

- The main catch here is that the ReLU function does not activate all the neurons at the same time.

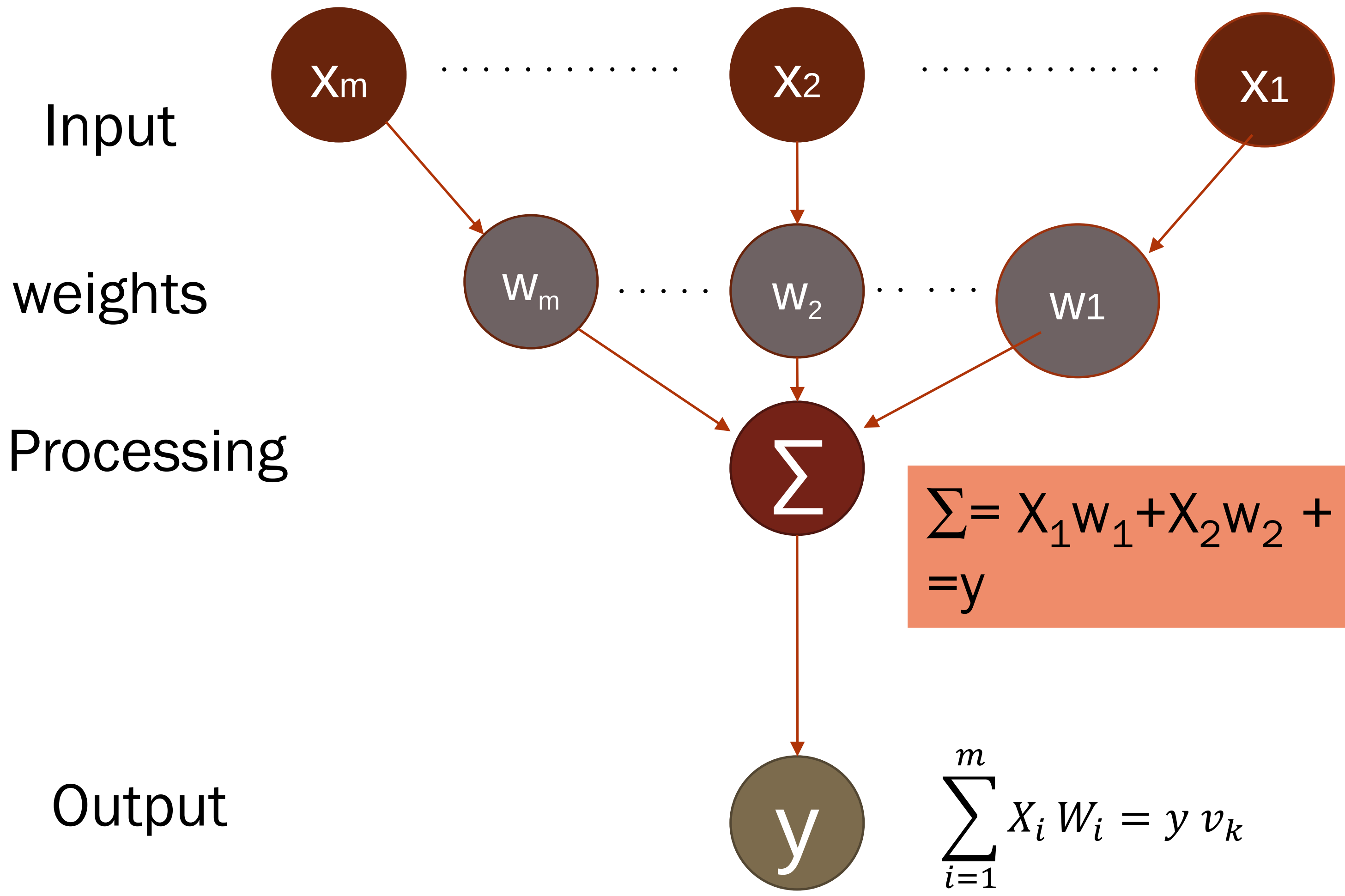
- The neurons will only be deactivated if the output of the linear transformation is less than 0.



Some other Non-linear Activation Functions of NN:



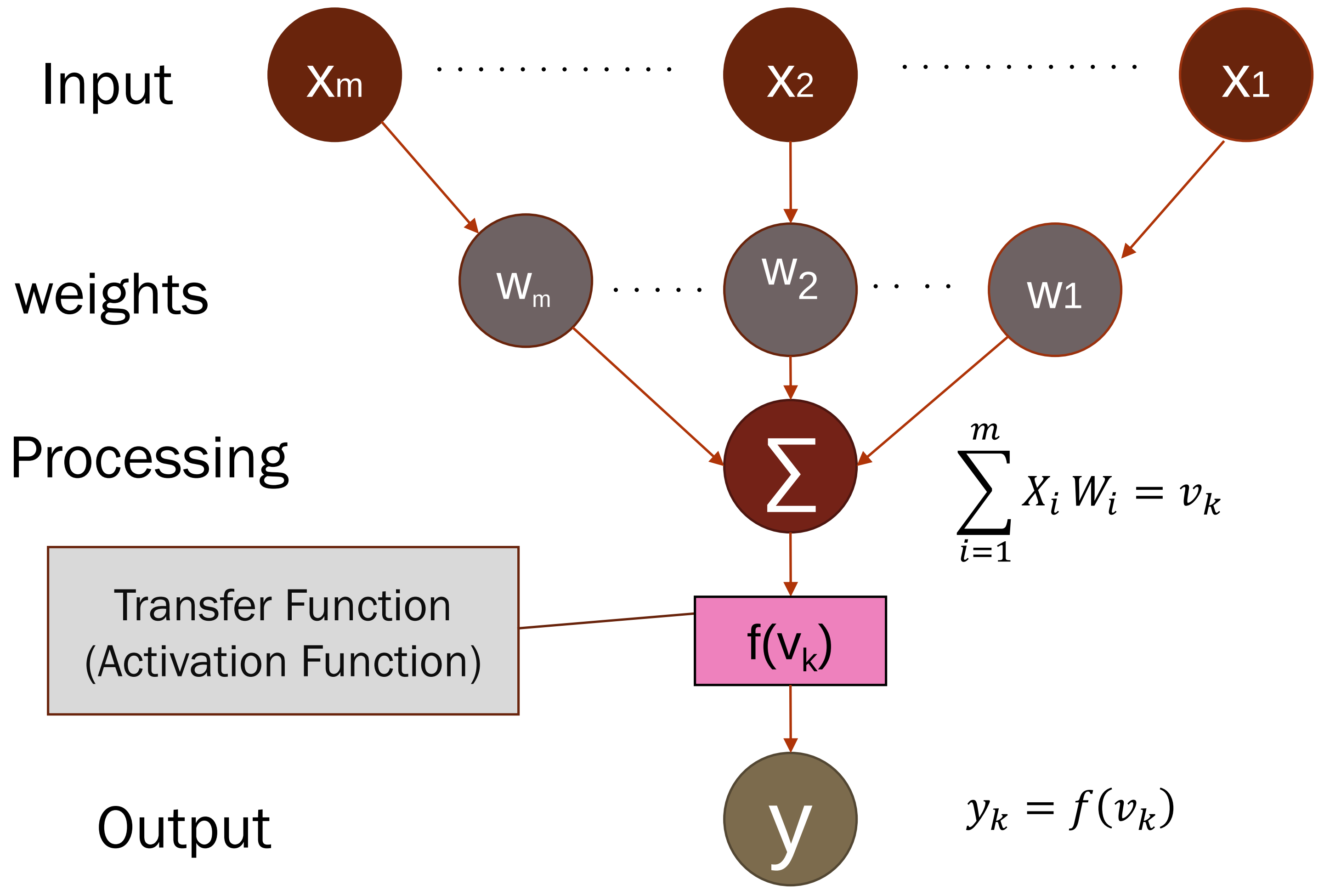
How do ANNs work?



$\sum = X_1w_1+X_2w_2 + \ldots.+X_mw_m$
$=y$

$$\sum_{i=1}^{m} X_i\, W_i = y\; v_k$$

How do ANNs work?



$$\sum_{i=1}^{m} X_i W_i = v_k$$

$$y_k = f(v_k)$$

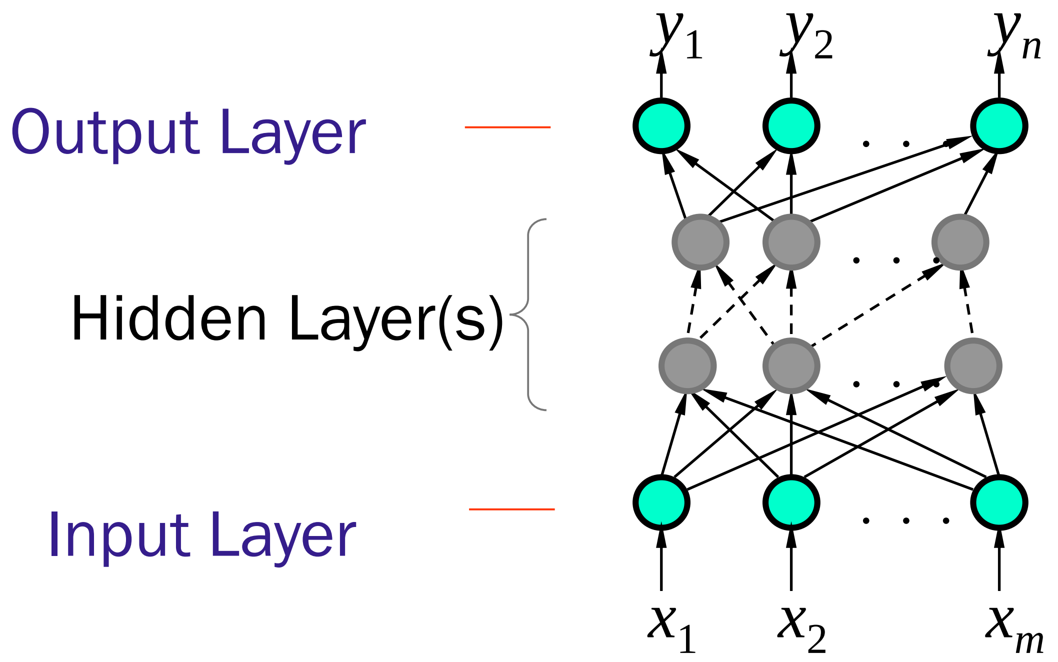
Feedforward vs. Backpropagation

- **Feedforward Propagation** - the flow of information occurs in the forward direction. The input is used to calculate some intermediate function in the hidden layer, which is then used to calculate an output.

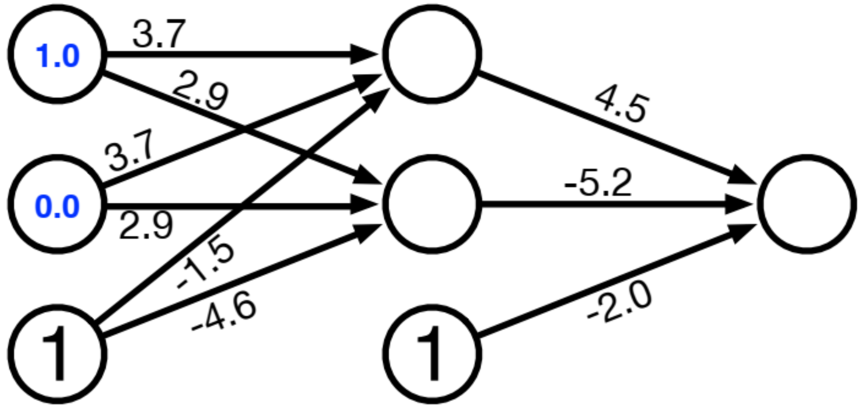
- **Backpropagation** - It aims to minimize the cost function by adjusting the network's weights and biases. The cost function gradients determine the level of adjustment with respect to parameters like activation function, weights, bias, etc.

Feed-Forward Neural Networks

- The feedforward neural network was the first and simplest type of artificial neural network devised. In this network, the information moves in only one direction—forward—from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network.
- Multi Layer Perceptron, - Two, Three, sometimes Four or Five Layers, But normally 3 layers are common structure.



Feed-Forward Neural Networks

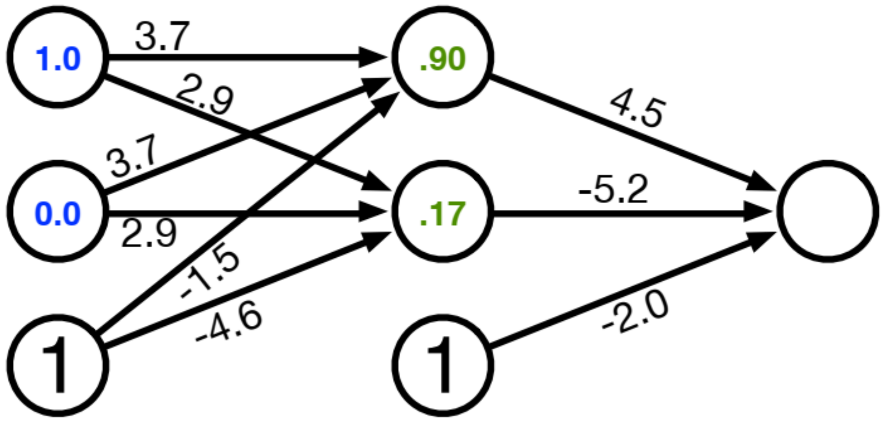


- Try out two input values
- Hidden unit computation

$$\text{sigmoid}(1.0 \times 3.7 + 0.0 \times 3.7 + 1 \times -1.5) = \text{sigmoid}(2.2) = \frac{1}{1 + e^{-2.2}} = 0.90$$

$$\text{sigmoid}(1.0 \times 2.9 + 0.0 \times 2.9 + 1 \times -4.5) = \text{sigmoid}(-1.6) = \frac{1}{1 + e^{1.6}} = 0.17$$

Feed-Forward Neural Networks

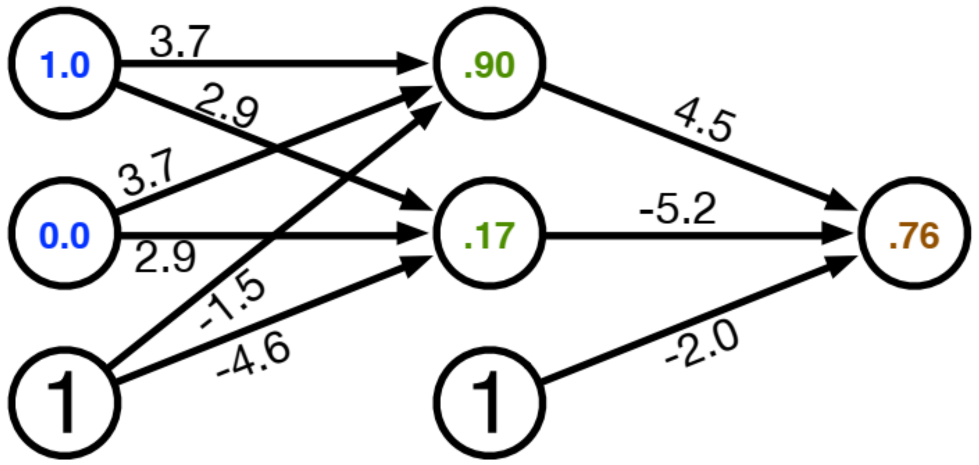


- Try out two input values
- Hidden unit computation

$$\text{sigmoid}(1.0 \times 3.7 + 0.0 \times 3.7 + 1 \times -1.5) = \text{sigmoid}(2.2) = \frac{1}{1 + e^{-2.2}} = 0.90$$

$$\text{sigmoid}(1.0 \times 2.9 + 0.0 \times 2.9 + 1 \times -4.5) = \text{sigmoid}(-1.6) = \frac{1}{1 + e^{1.6}} = 0.17$$

Feed-Forward Neural Networks



- Output unit computation

$$\text{sigmoid}(.90 \times 4.5 + .17 \times -5.2 + 1 \times -2.0) = \text{sigmoid}(1.17) = \frac{1}{1 + e^{-1.17}} = 0.76$$

What is a backpropagation

- Backpropagation, or backward propagation of errors, is an algorithm that is designed to test for errors working back from output nodes to input nodes. It is an important mathematical tool for improving the accuracy of predictions in data mining and machine learning.
- Essentially, backpropagation is an algorithm used to calculate derivatives quickly.

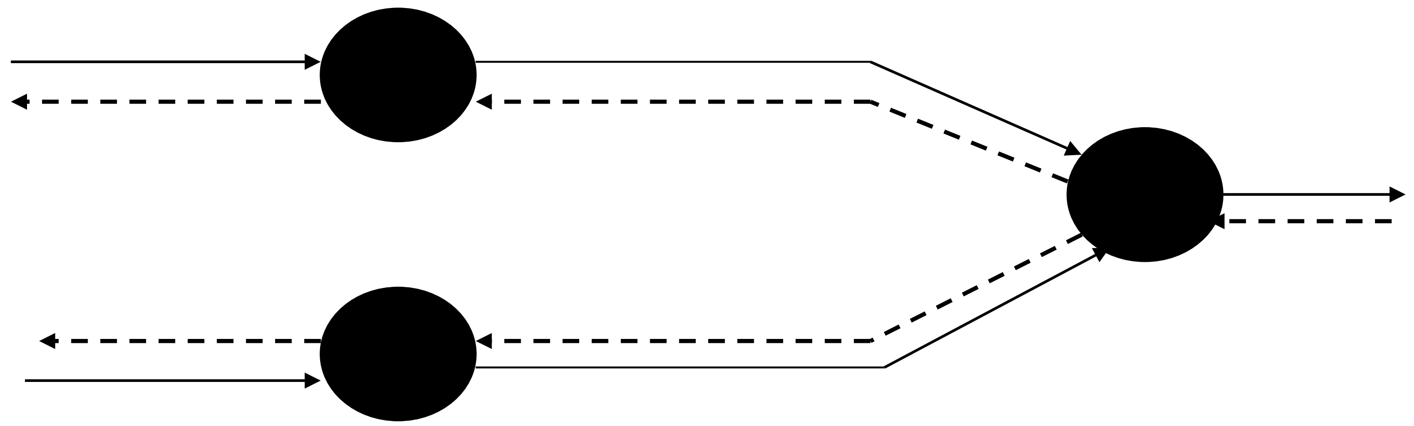


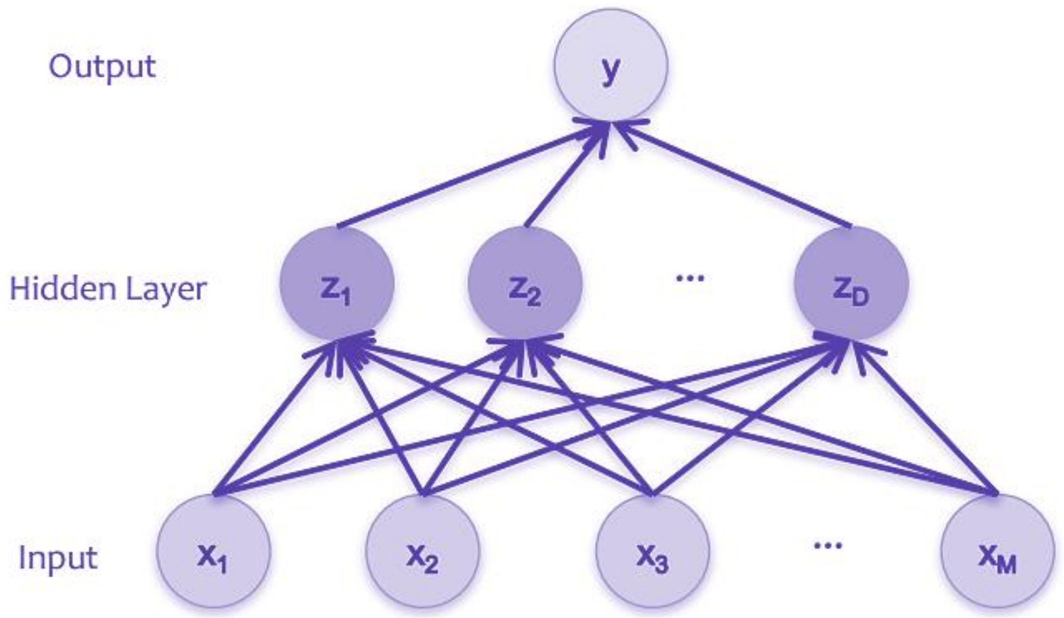
Fig: Signal Flow Backpropagation of Errors

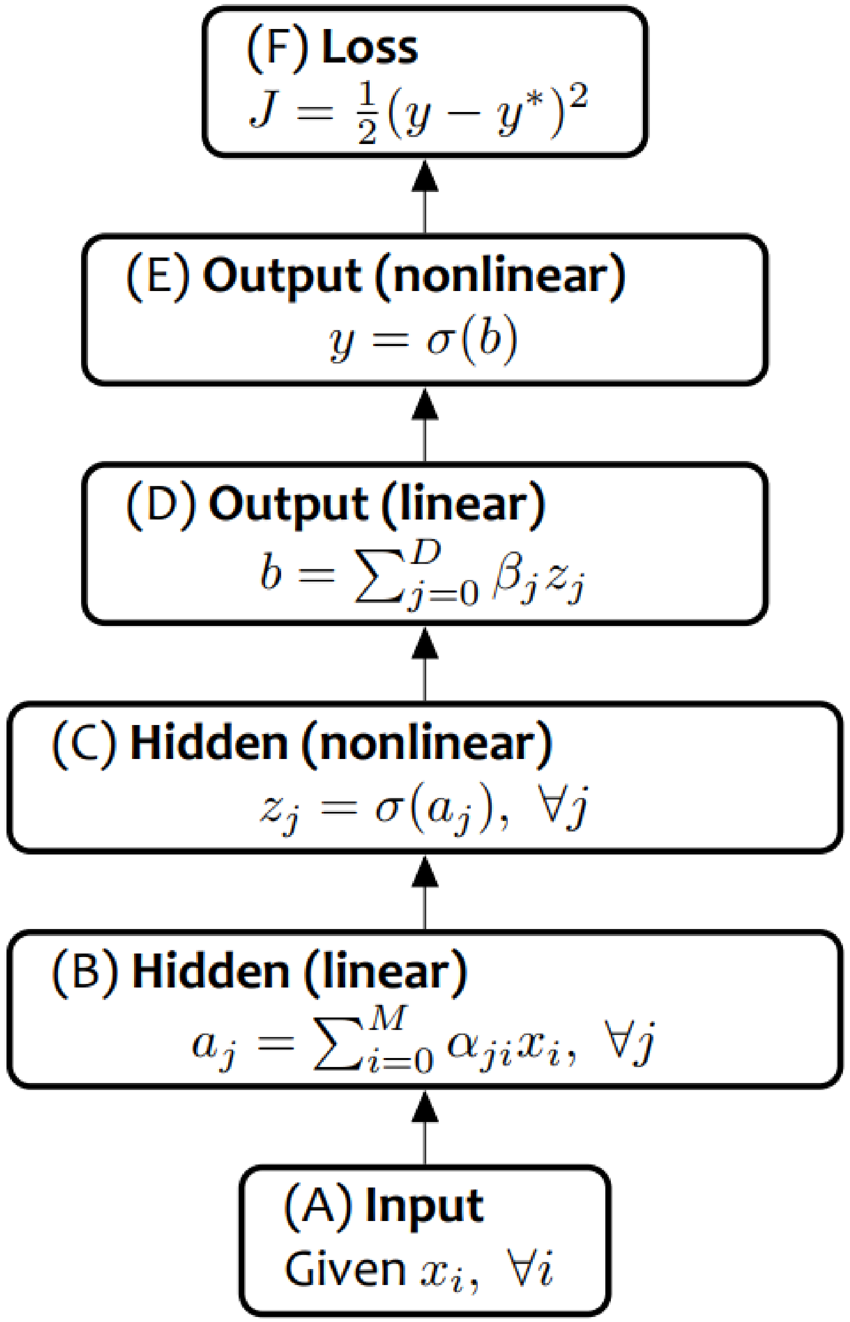
→ **Function Signals**

⇠ **Error Signals**

ANN

- Artificial neural network




(F) Loss
$J = \frac{1}{2}(y - y^*)^2$
(E) Output (nonlinear)
$y = \sigma(b)$
(D) Output (linear)
$b = \sum_{j=0}^{D} \beta_j z_j$
(C) Hidden (nonlinear)
$z_j = \sigma(a_j), \ \forall j$
(B) Hidden (linear)
$a_j = \sum_{i=0}^{M} \alpha_{ji} x_i, \ \forall j$
(A) Input
Given $x_i, \ \forall i$

Backpropagation Algo - Chain Rule

- The chain rule allows us to find the derivative of a composite function. The composite function, h = g(f(x)), then its derivative as given by the chain rule is:

$$\frac{dh}{dx} = \frac{dh}{du} \cdot \frac{du}{dx}$$

Here, u is the output of the inner function f (hence, u = f(x)), which is then fed as input to the next function g to produce h (hence, h = g(u)).

- The generalize chain rule: $\frac{\partial h}{\partial x_i} = \frac{\partial h}{\partial u_1} \cdot \frac{\partial u_1}{\partial x_i} + \frac{\partial h}{\partial u_2} \cdot \frac{\partial u_2}{\partial x_i} + \ldots + \frac{\partial h}{\partial u_n} \cdot \frac{\partial u_n}{\partial x_i}$

$$\frac{\partial h}{\partial x_i} = \sum_j \frac{\partial h}{\partial u_j} \cdot \frac{\partial u_j}{\partial x_i}$$

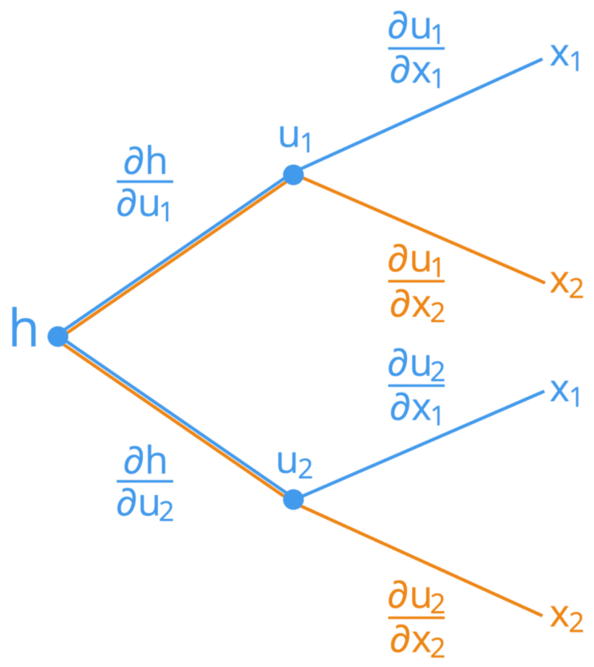
where j = 1, …, n; i = 1, …, m

Backpropagation Algo - Chain Rule

- Suppose that we have a composite function of two independent variables, x1 and x2, defined as follows:

$$h = g(f(x1, x2)) = g(u1(x1, x2), u2(x1, x2))$$

Here, u1 and u2 act as the intermediate variables.



h
u_1
u_2
x_1
x_2
x_1
x_2
$\frac{\partial h}{\partial u_1}$
$\frac{\partial h}{\partial u_2}$
$\frac{\partial u_1}{\partial x_1}$
$\frac{\partial u_1}{\partial x_2}$
$\frac{\partial u_2}{\partial x_1}$
$\frac{\partial u_2}{\partial x_2}$

$$\boxed{\begin{aligned} \frac{\partial h}{\partial x_1} &= \frac{\partial h}{\partial u_1} \cdot \frac{\partial u_1}{\partial x_1} + \frac{\partial h}{\partial u_2} \cdot \frac{\partial u_2}{\partial x_1} \\ \frac{\partial h}{\partial x_2} &= \frac{\partial h}{\partial u_1} \cdot \frac{\partial u_1}{\partial x_2} + \frac{\partial h}{\partial u_2} \cdot \frac{\partial u_2}{\partial x_2} \end{aligned}}$$

Notice how the chain rule relates the net output, *h*, to each of the inputs, x_i, through the intermediate variables, u_j. This is a concept that the backpropagation algorithm applies extensively to optimize the weights of a **neural network.**

Backpropagation NN - Example

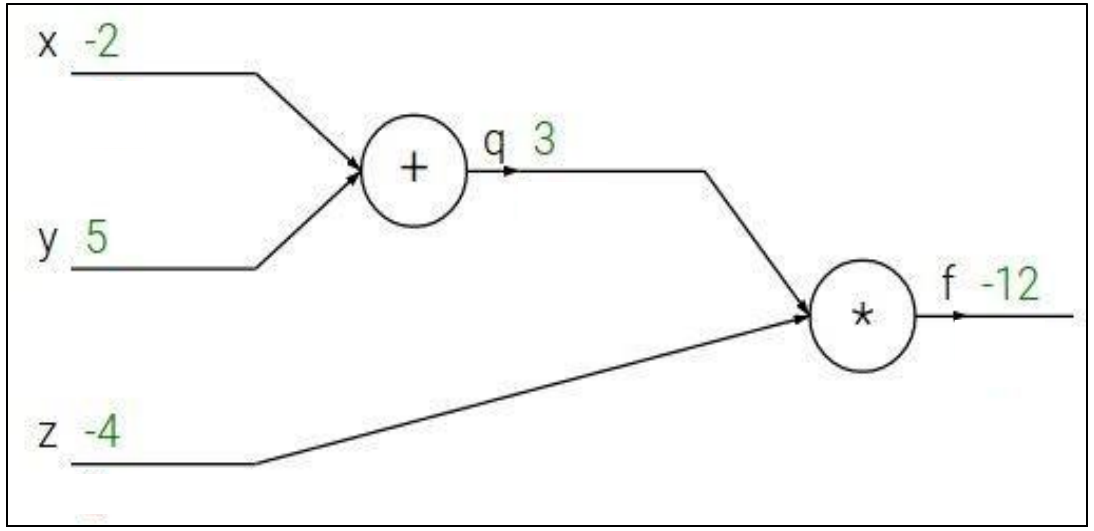
- E.g: $x = -2$, $y = 5$, $z = -4$

$$f(x,y,z) = (x+y)z$$

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want to find: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Backpropagation NN - Example

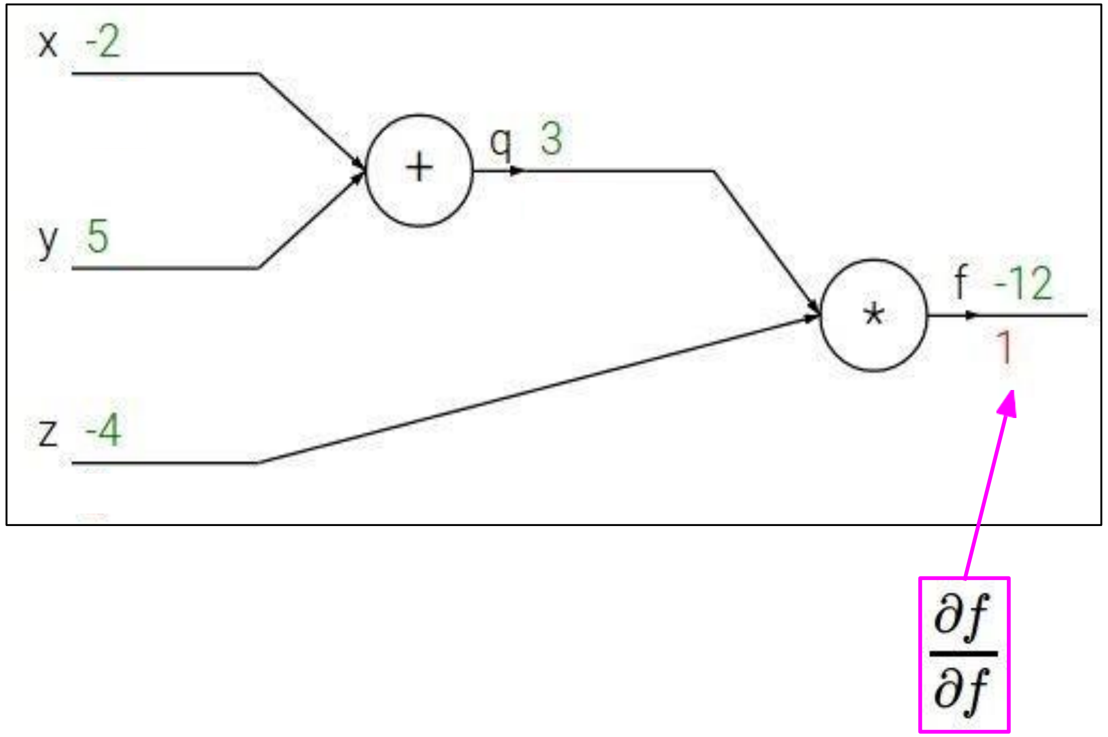
- E.g: x = -2, y = 5, z = -4

$$f(x, y, z) = (x + y)z$$

$$\boxed{q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1}$$

$$\boxed{f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q}$$

Want to find: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\boxed{\frac{\partial f}{\partial f}}$$

Backpropagation NN - Example

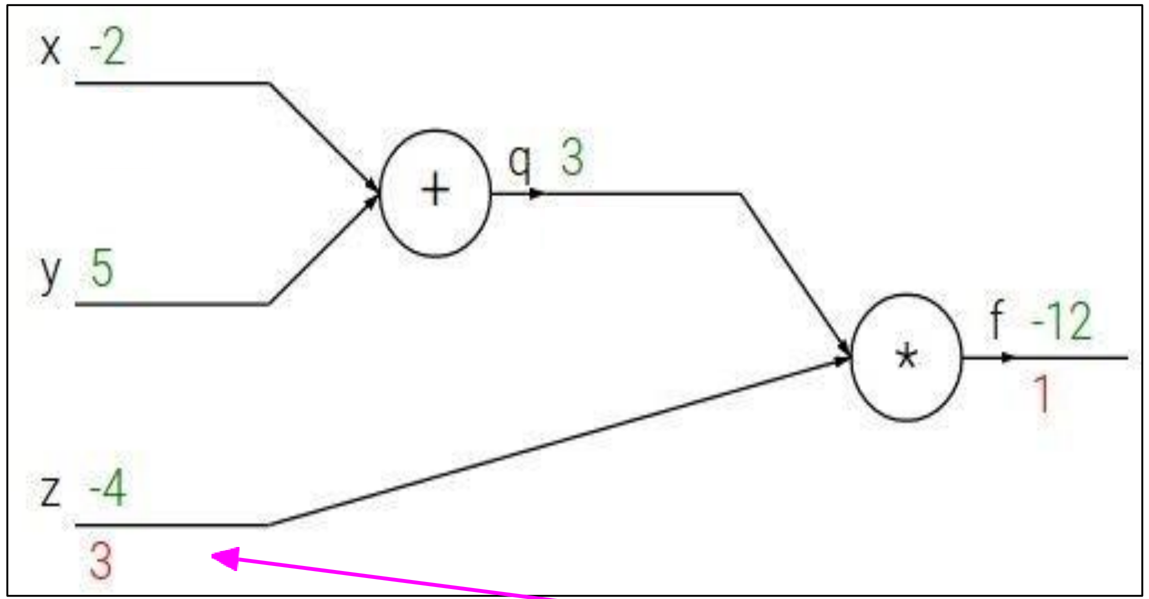
- E.g: x = -2, y = 5, z = -4

$$f(x,y,z) = (x+y)z$$

$$\boxed{q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1}$$

$$\boxed{f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q}$$

Want to find: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\boxed{\frac{\partial f}{\partial z}}$$

Backpropagation NN - Example

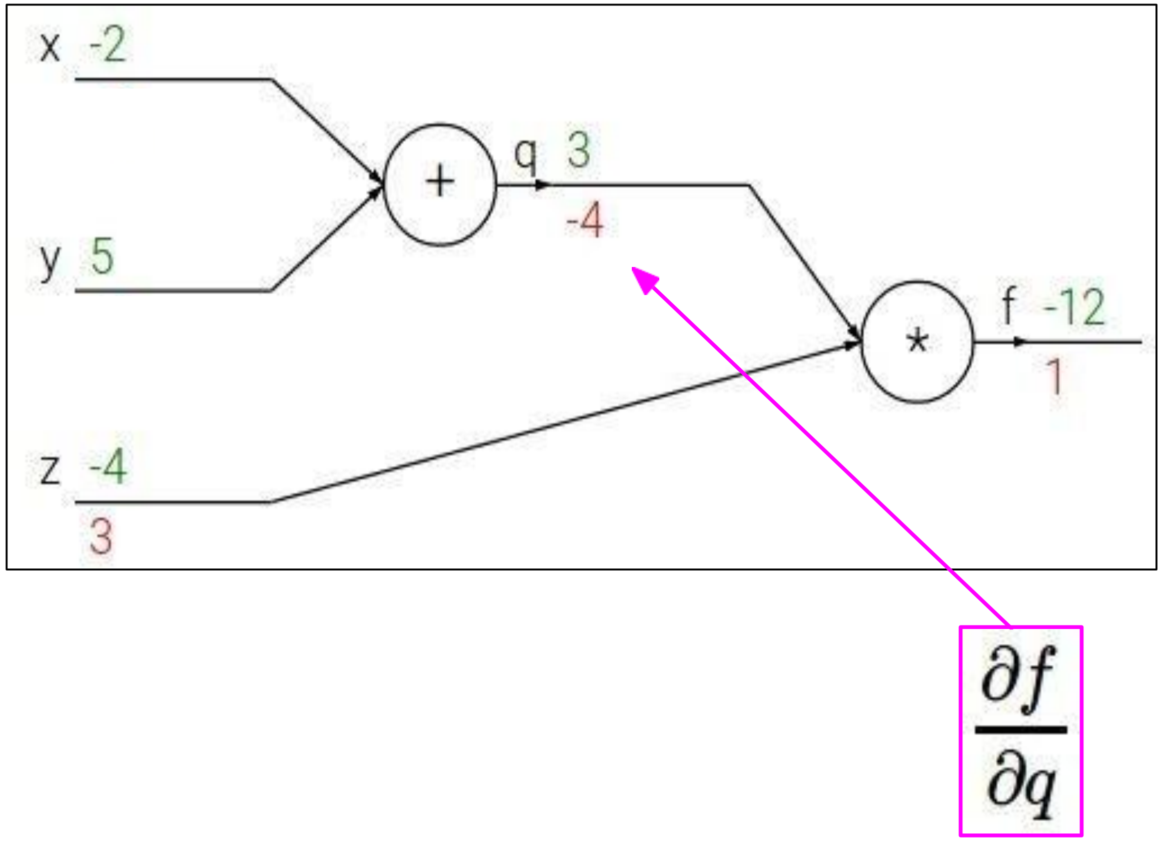
- E.g: x = -2, y = 5, z = -4

$$f(x,y,z) = (x+y)z$$

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want to find: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Backpropagation NN - Example

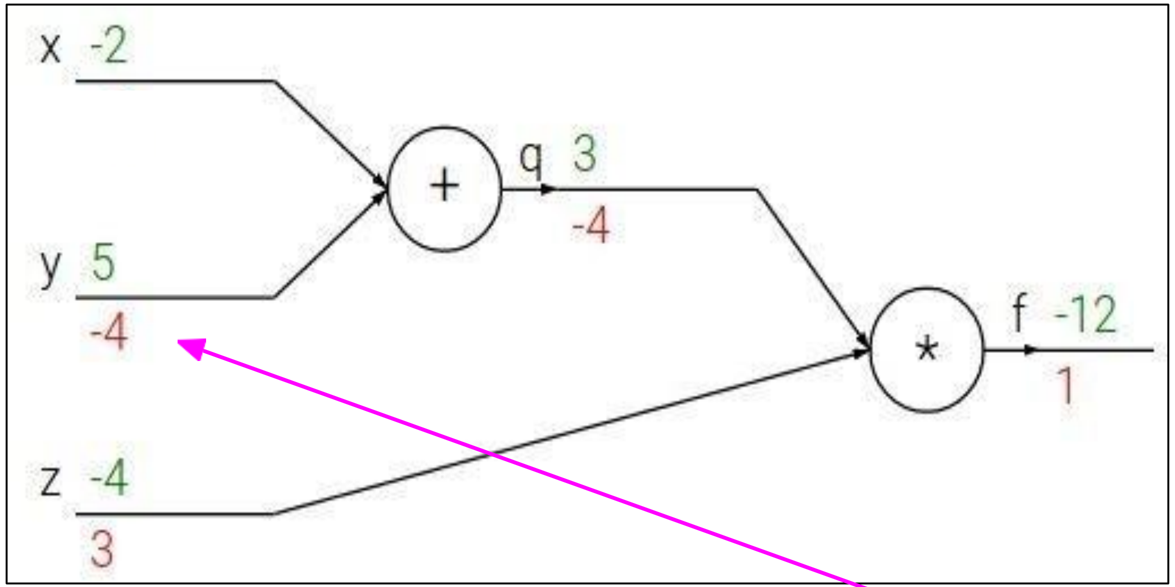
- E.g: x = -2, y = 5, z = -4

$$f(x,y,z) = (x+y)z$$

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want to find: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial y}$$

Chain rule:

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial y}$$

Backpropagation NN - Example

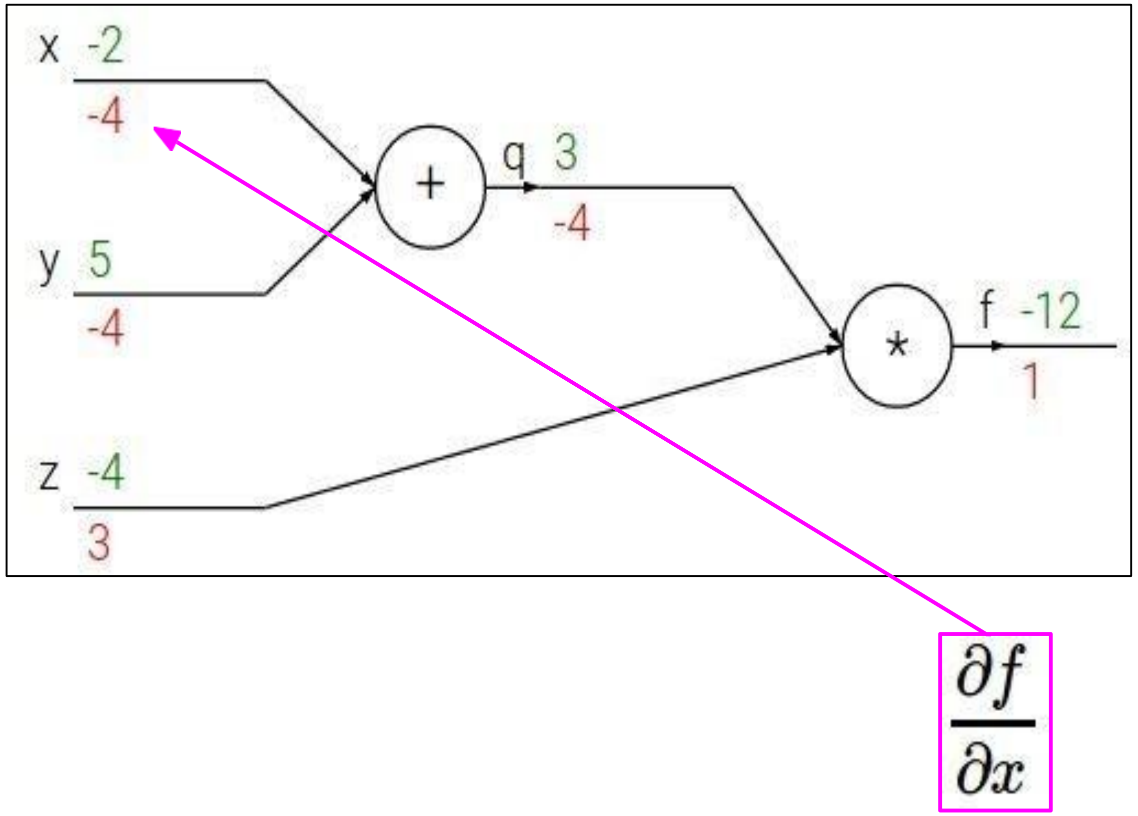
- E.g: x = -2, y = 5, z = -4

$$f(x,y,z) = (x+y)z$$

$$\boxed{q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1}$$

$$\boxed{f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q}$$

Want to find: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Chain rule:

$$\boxed{\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial y}}$$

Thank you