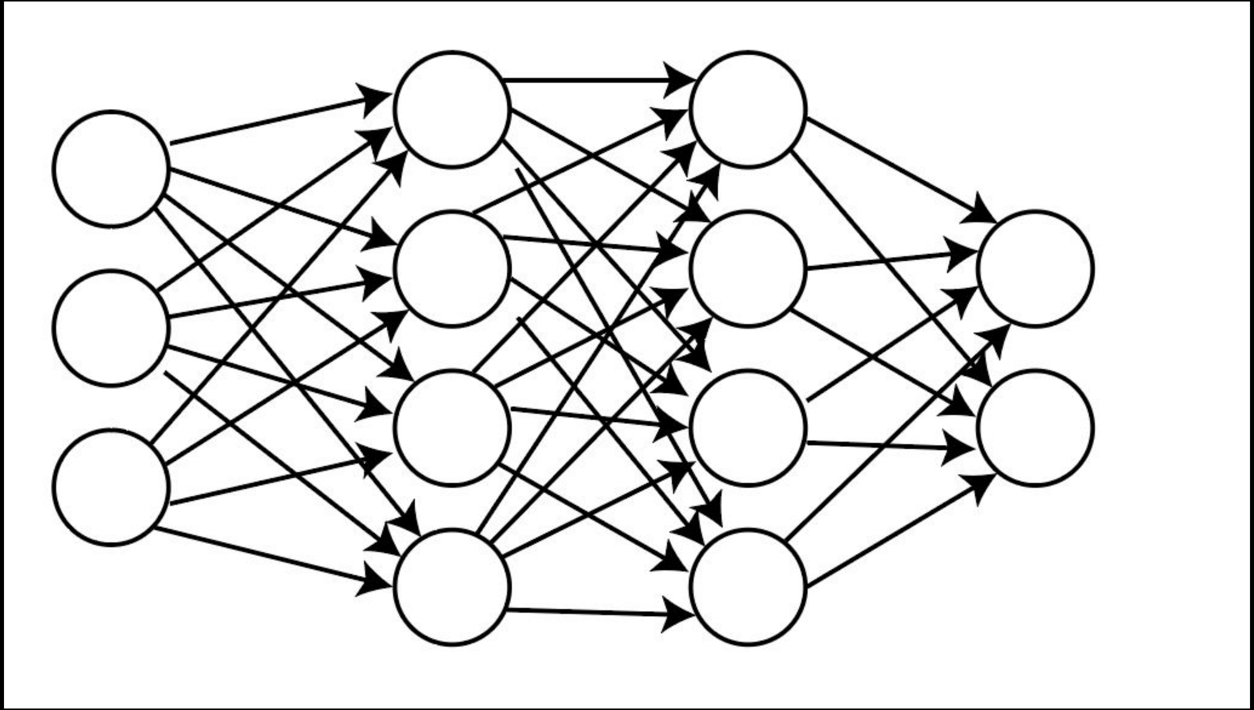
Multilayer Perceptrons



Perceptrons

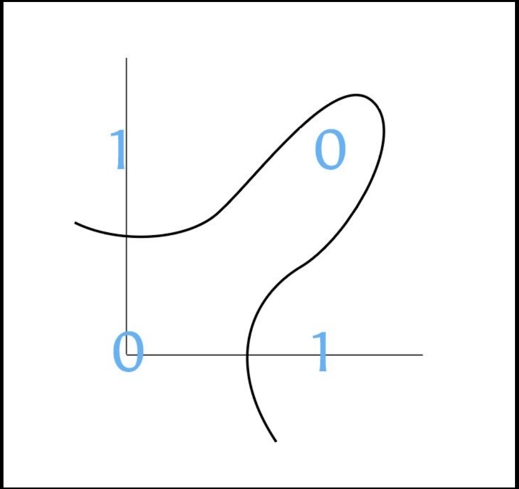
Developed in the late 50s by Frank Rosenblatt

- Initially a physical machine
  - Designed for image recognition
  - Used an array of 400 photocells, connected randomly to '*neurons*'
  - *Weights* were encoded with potentiometers
  - *Weight updates* and *learning* performed by electric motors

Perceptrons

Developed in the late 50s by Frank Rosenblatt

- Limitations
  - The design seemed promising, but unable to be trained on many types of patterns
    - Unable to be trained to learn an *XOR* function which are not linearly separable
  - The inability to work on functions that aren't linearly separable lays in its single layer design



Neuron

What is a 'Neuron?'

A *Node, or Neuron* is a single unit in a *Multilayer Perceptron*

Functions similarly to a neuron in the brain

- a Multilayer Perceptron is a type of *Neural Network*

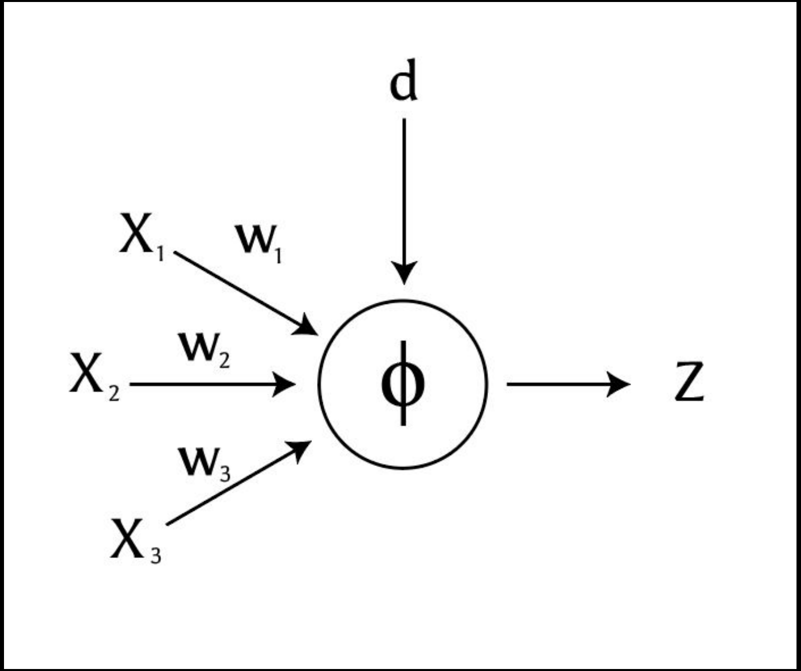
Neuron Structure

Inputs:

- Numerical or Vector Inputs
    - With their corresponding Weights
- Bias term

Outputs:

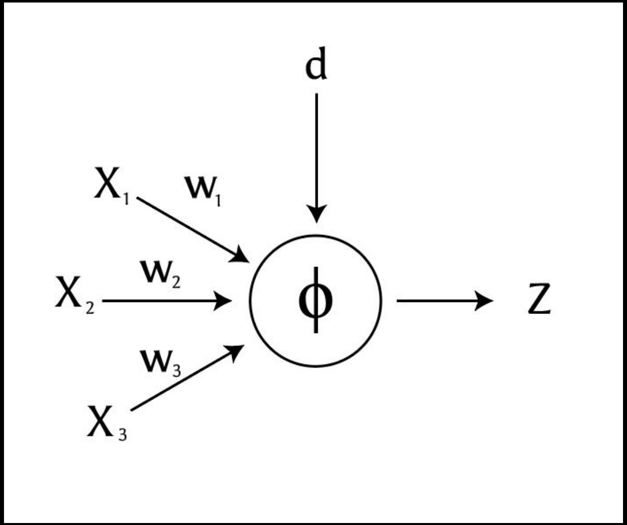
- An Activation Value

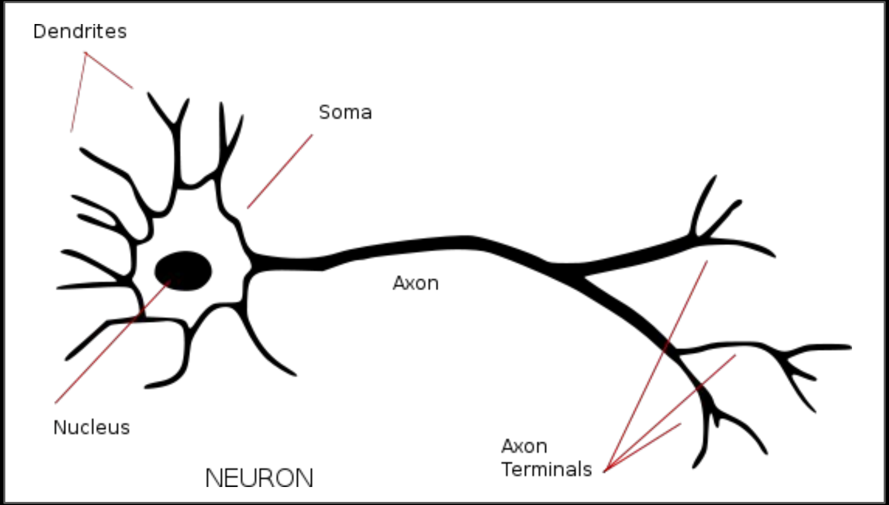


Neuron Structure

The inputs and bias are similar to the terminals of a biological neuron, which then requires a certain strength of incoming signal to be *activated* and send a signal to further neurons.

MLPs learn by iteratively adjusting *weights* to fit the given problem.

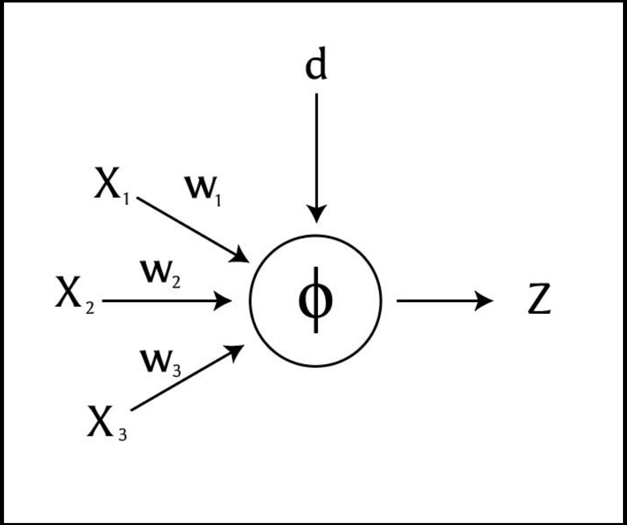


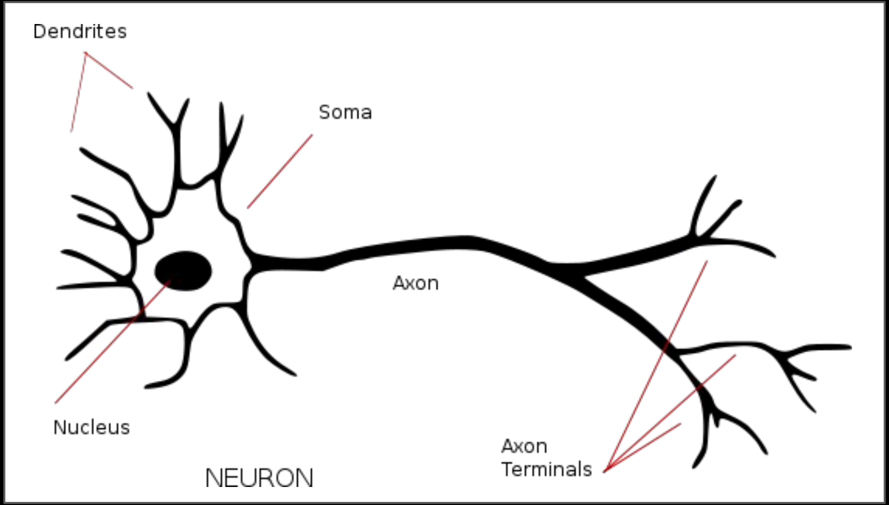


Neuron Structure

For metaphorical purposes, connections of neurons ‘learn’ by them weighting their incoming signals given different tasks.

When a certain task appears in the future, the learned weights allow the neurons to easily process inputs and create an output.



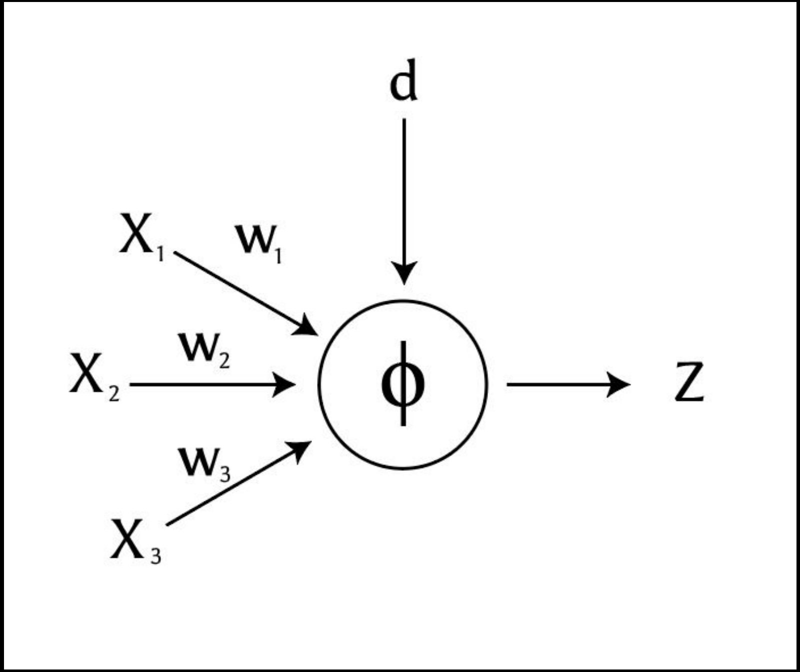


Math behind a Neuron

Enough of the history and metaphors, let’s dig into the numbers.

Elements of a Perceptron

- Inputs: x_i
- Weights w_i
- Bias: b
- Activation Function: $\phi(z)$
- Output: Z



Math Behind a Neuron

$$\phi\left(b + \sum_{i=1}^{n} x_i \cdot w_i\right) = Z$$

1. The dot product of each of the weights and the inputs are summed
2. The bias term is added to the summed activation
3. The activation function is applied to the result

One Hot Encoding

One Hot Encoding

Let’s take a step back to our inputs for a moment.

- Our perceptrons function off of multiplication or vector multiplication.
- But not all incoming data is numerical.
    - I.e. Male or Female, Country of Origin, Breed, etc.

Given an MLP requires numeric data, we require *one hot encoding* to encode these categorical variables into numerical variables our nodes can pass around.

One Hot Encoding Mechanics

For a given categorical variable, each of the categories are assigned a binary variable.

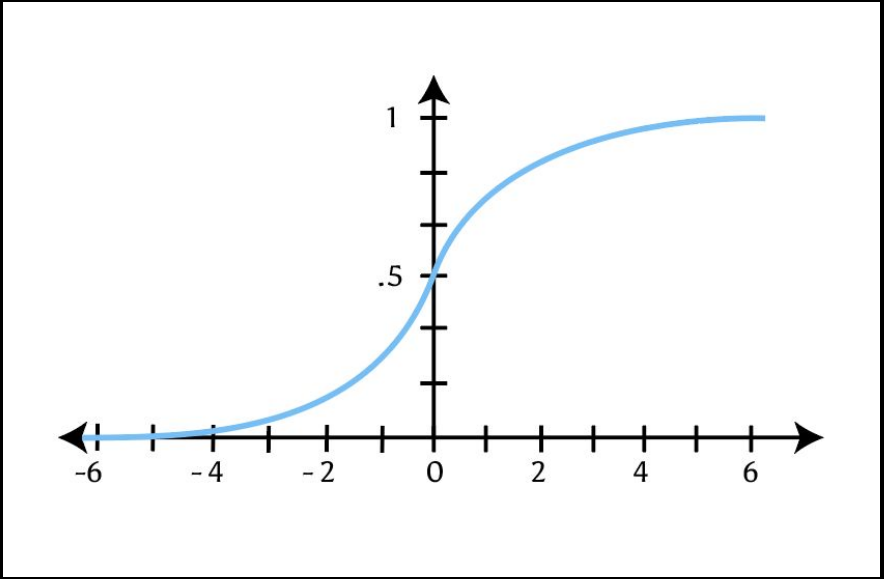
- Let's say we had a variable for the color of each students class notebook in a given lecture.
    - The only available colors at this school are red, yellow, and blue.
- In the process of one hot encoding, we'll encode each of these categories into a variable.

Activation Functions

Sigmoid Activation Function

Outputs between 0 and 1 (inclusive)

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



ReLu Activation Function

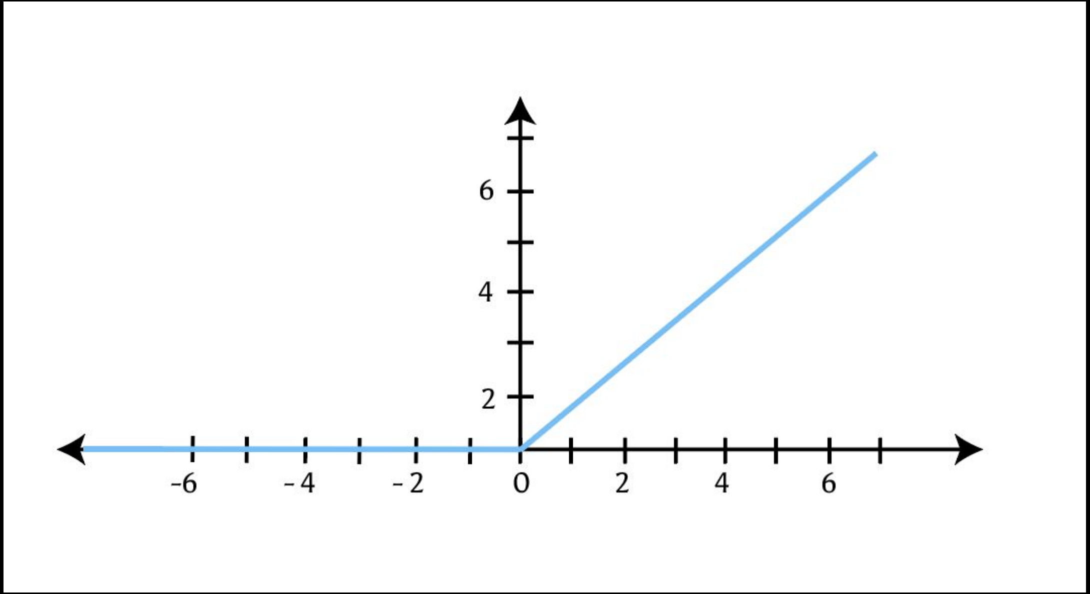
Code:

$$ReLu(x) = max(0, x)$$

Piecewise Function:

$$ReLu(x) = \begin{cases} 0 & x \leq 0 \\ x & x > 0 \end{cases}$$

Outputs from 0 to ∞ (inclusive)



Layers

Stacking Perceptrons a.k.a. Layers

If we line up Neurons, we form a *Layer*, more specifically, a *Hidden Layer*

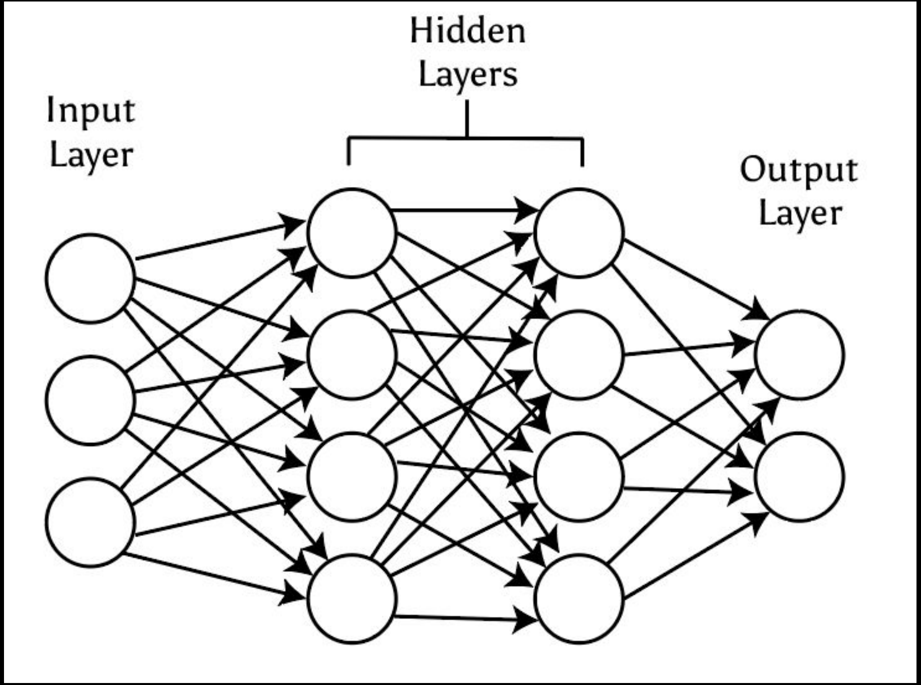
Hidden layers perform weighted sums modified by the activation function on the weights of the previous layer

Layer Types

- Input
  - Corresponds to the columns of the input data
- Hidden
  - Contain the perceptron nodes that
  - Earlier layers transform the input into more linearly separable tasks for later layers
- Output
  - Corresponds to the output classes
    - Binary, Multiclass, etc.

Overall Structure of a Perception

Perceptrons are *fully connected,* meaning that each node is connected to every node from the previous and following layer (if it exists)



Training the MLP - next time!

- Gradient descent
- Error functions
- Backpropagation