Deep Learning

Course Introduction

Scope of the Course

Course Duration

12-25 Jan

8 Sessions, 2 Labs, ***26 Hours in Total***

Structure of Course

What **Why** **When** **Where** **How**

~~Who~~

Artificial Neural Network → Convolutional Neural Network

Course Projects

2 Mini Projects & Assignments

Ass-1 **ANN** (Churn Prediction For Bank Customer),
Ass-2 **CNN** (Image Classification),

1 Medium Projects

CNN (Image (dog/cat) Classification),

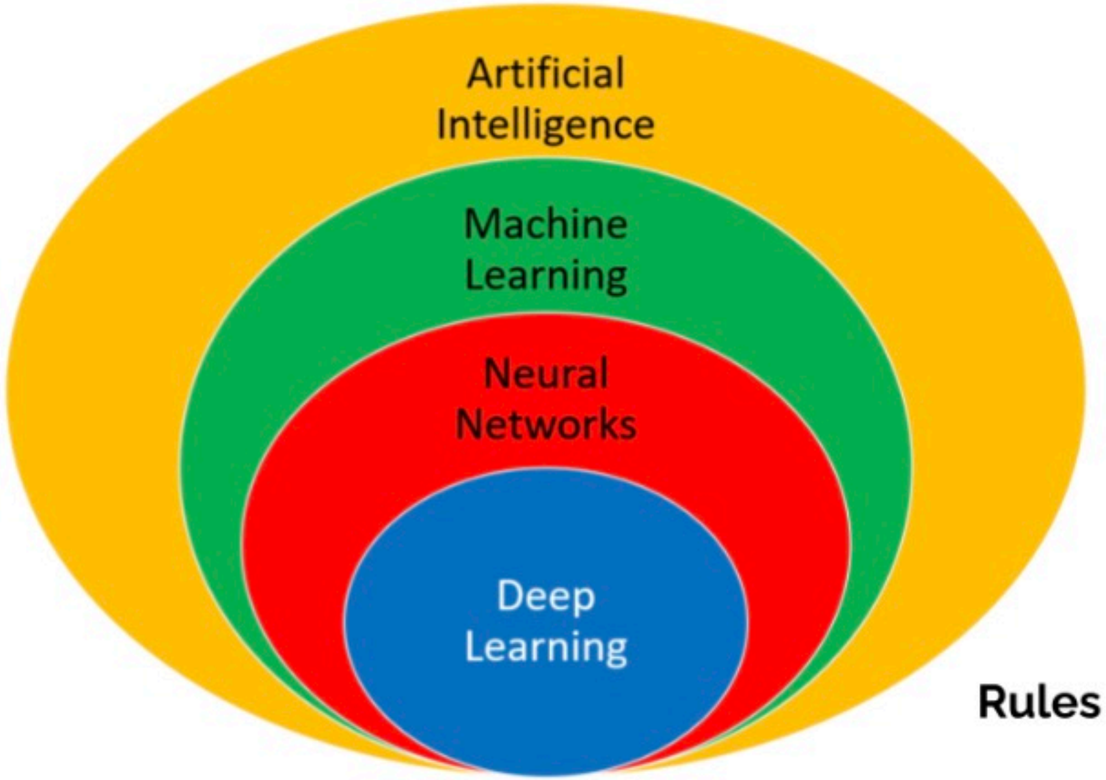
Deep Learning

Session-1

Table of Contents

What is Deep Learning?

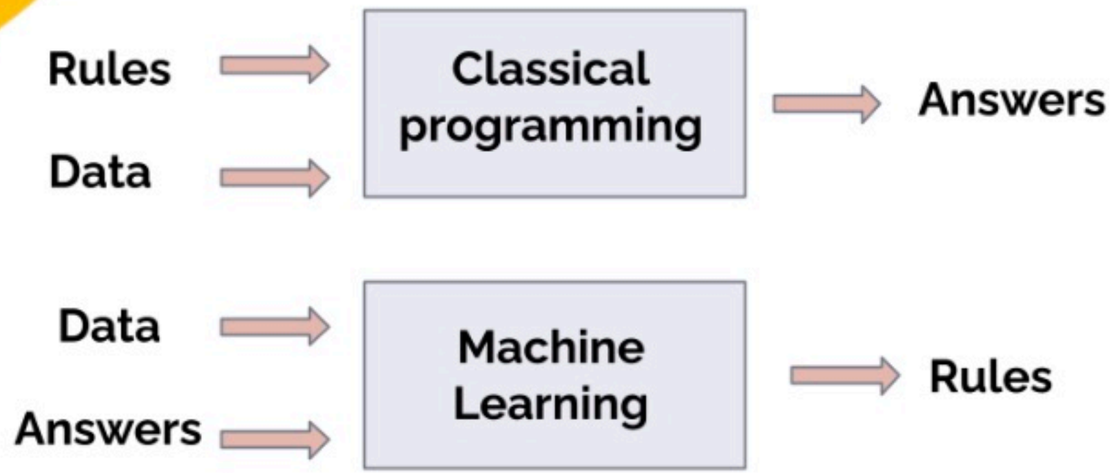
What is Deep Learning?



AI

The effort to automate intellectual tasks normally performed by humans. AI is a general field that encompasses machine learning and deep learning.

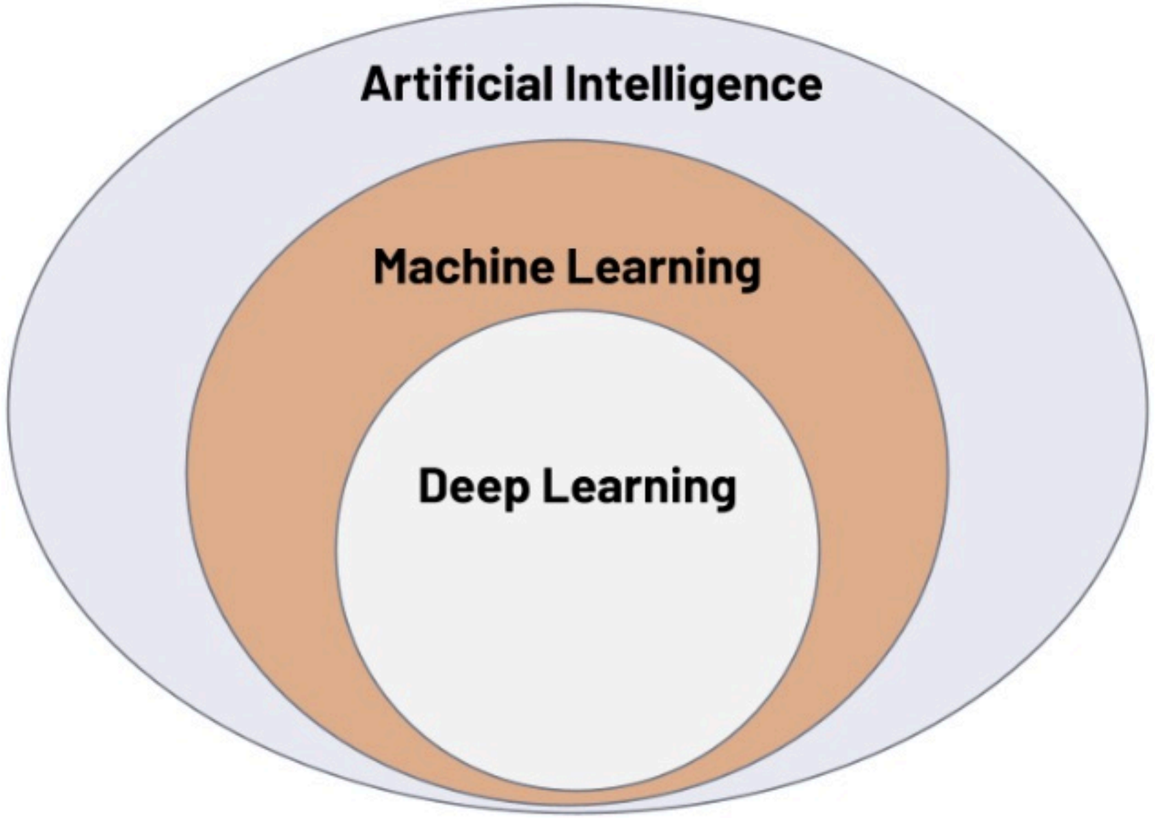
ML/DL

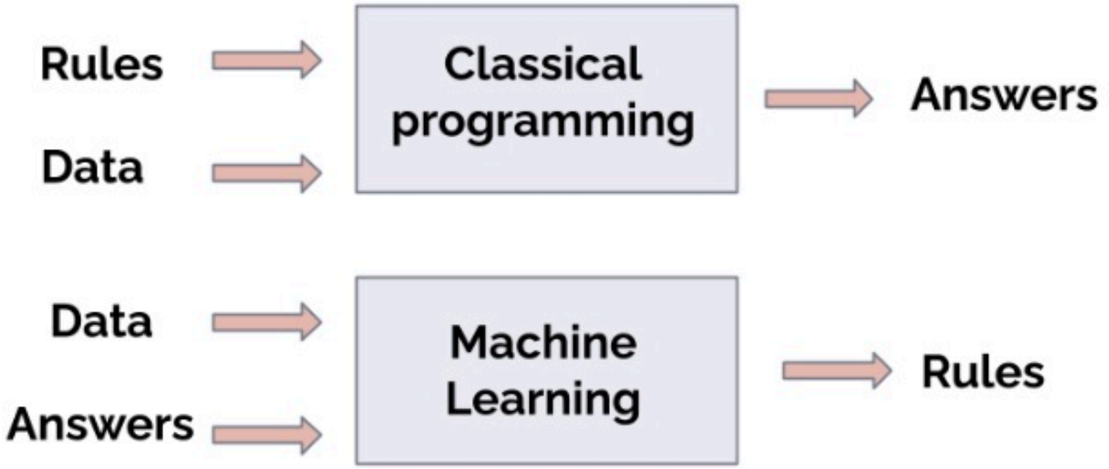


"the field of study that gives computers the ability to learn without being explicitly programmed."

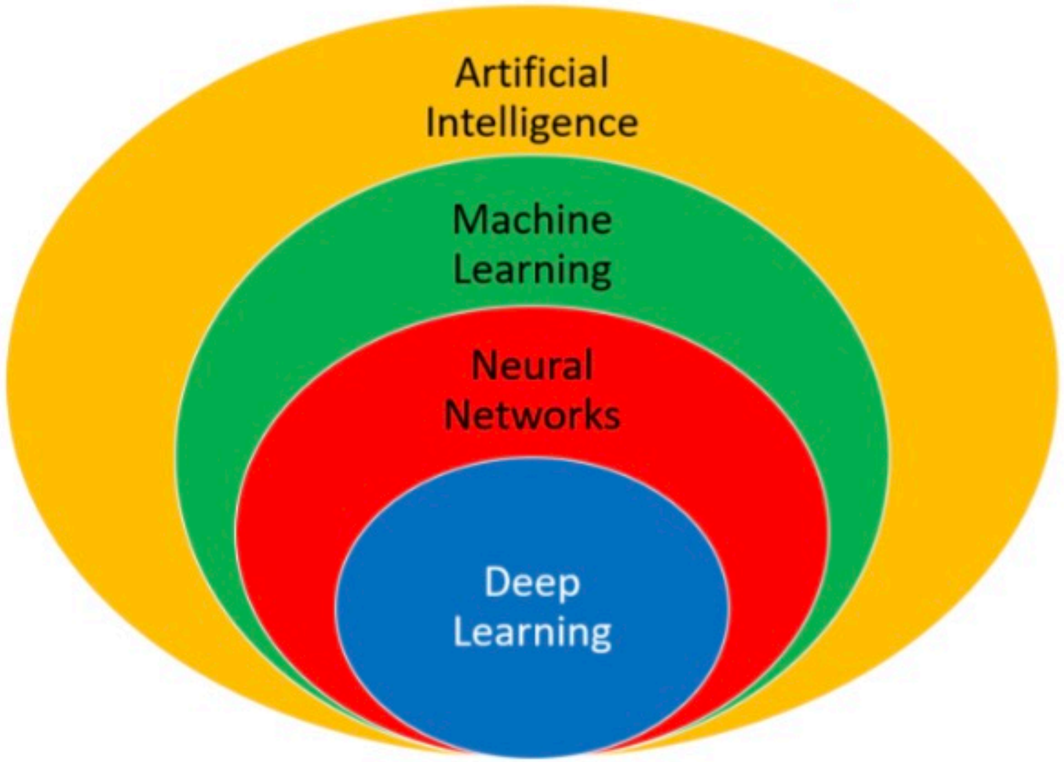
A new programming paradigm

What is Deep Learning?





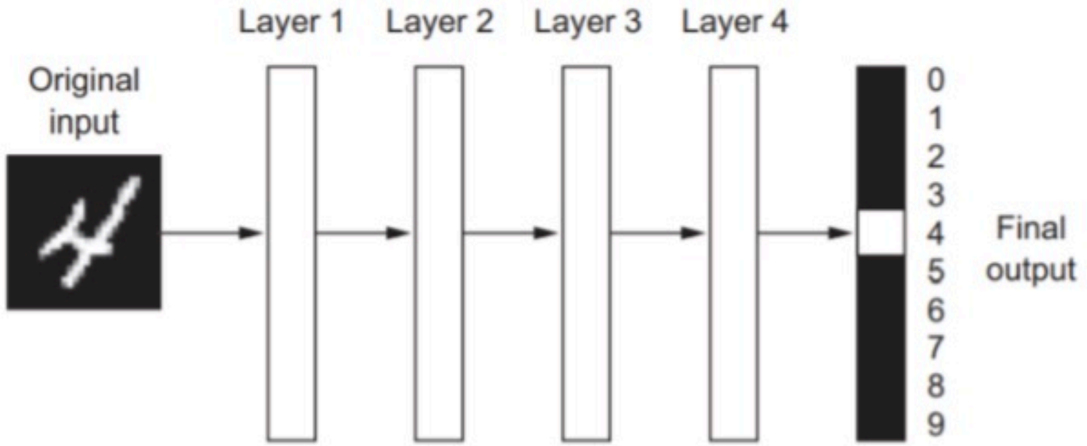
What is Deep Learning?



Deep Learning

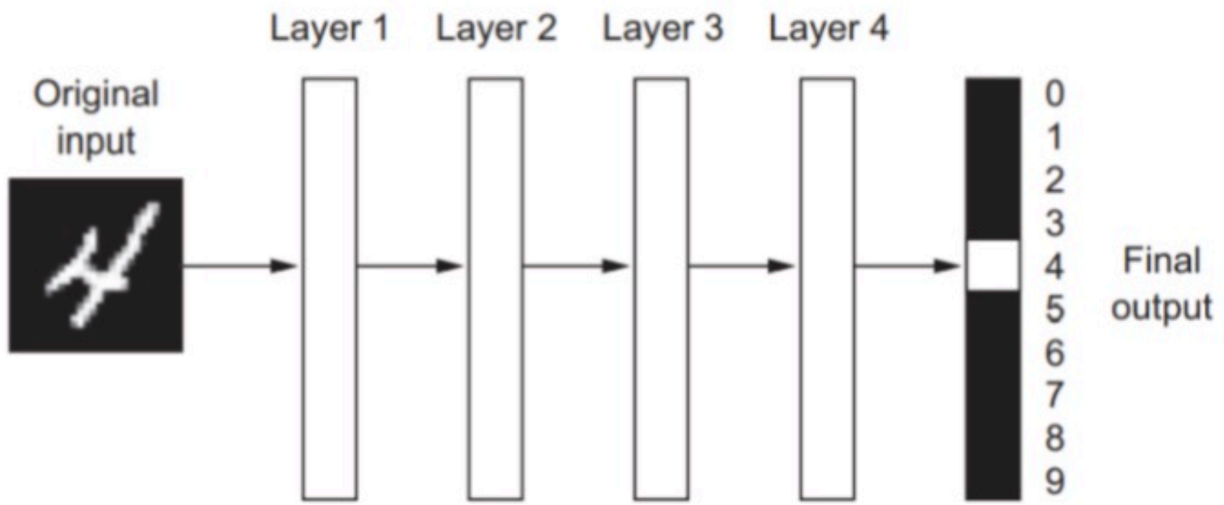
***Subfield** of machine learning concerned with algorithms **inspired by the structure and function of the brain** called **artificial neural networks**.*

"The deep in deep learning isn't a reference to any kind of deeper understanding achieved by the approach; rather, it stands for this idea of successive layers of representations"



What is Deep Learning?

***Subfield** of machine learning concerned with algorithms **inspired by the structure and function of the brain** called **artificial neural networks**.*



What is Deep Learning?

Andrew Ng

The idea of deep learning as:

Using brain simulations, hope to:

– *Make learning algorithms **much better and easier** to use.*

– *Make **revolutionary advances** in machine learning and AI.*

*I believe this is our best shot at progress towards **real AI***

*..almost all the value today of deep learning is through **supervised learning** or learning from labeled data*

*..one reason that deep learning has taken off like crazy is because it is fantastic at **supervised learning***

What is Deep Learning?

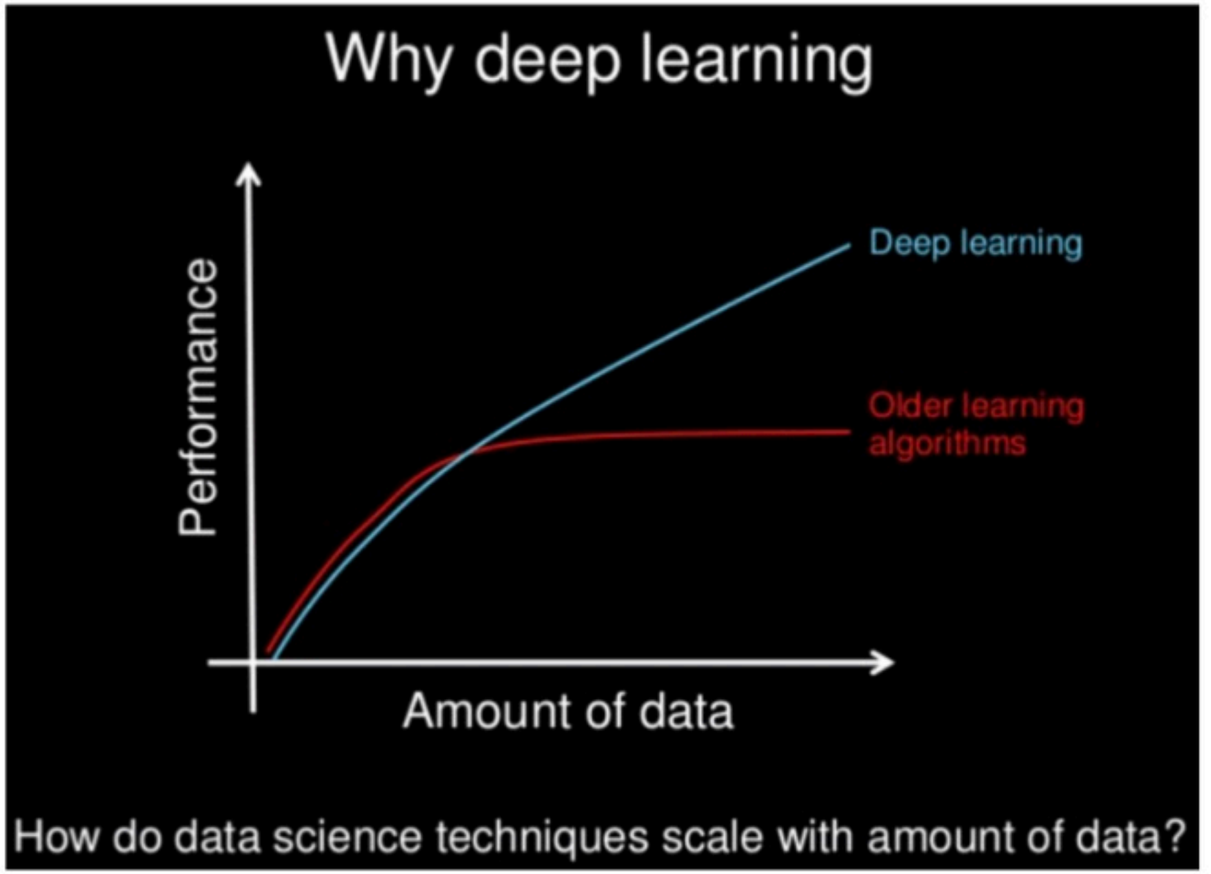
Scalability

very large neural networks *we can now have and …* ***huge amounts of data*** *that we have access to*

for most flavors of the ***old generations of learning algorithms*** *…* ***performance will plateau****.*

…

deep learning *… is the first class of algorithms … that is scalable. …* ***performance just keeps getting better as you feed them more data***

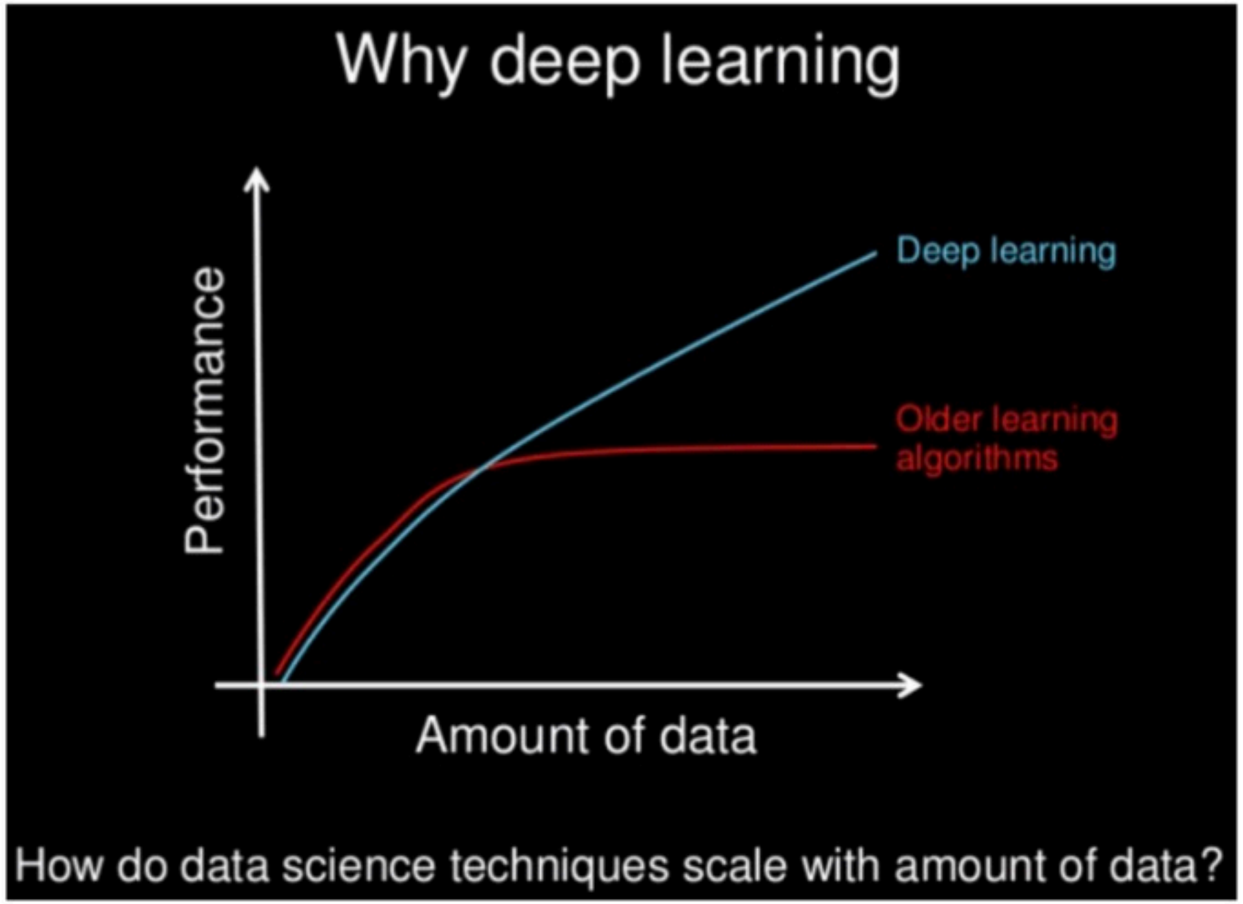


Why Deep Learning?
Slide by Andrew Ng, all rights reserved.

What is Deep Learning?

Scalability

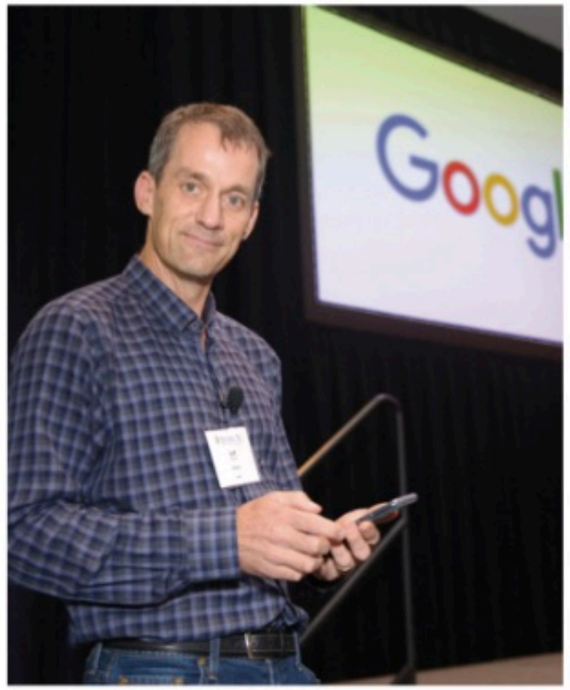
very large neural networks *we can now have and …* ***huge amounts of data*** *that we have access to*



Why Deep Learning?
Slide by Andrew Ng, all rights reserved.

What is Deep Learning?

Scalability



Jeff Dean

Important Property of Neural Networks

Results get better with

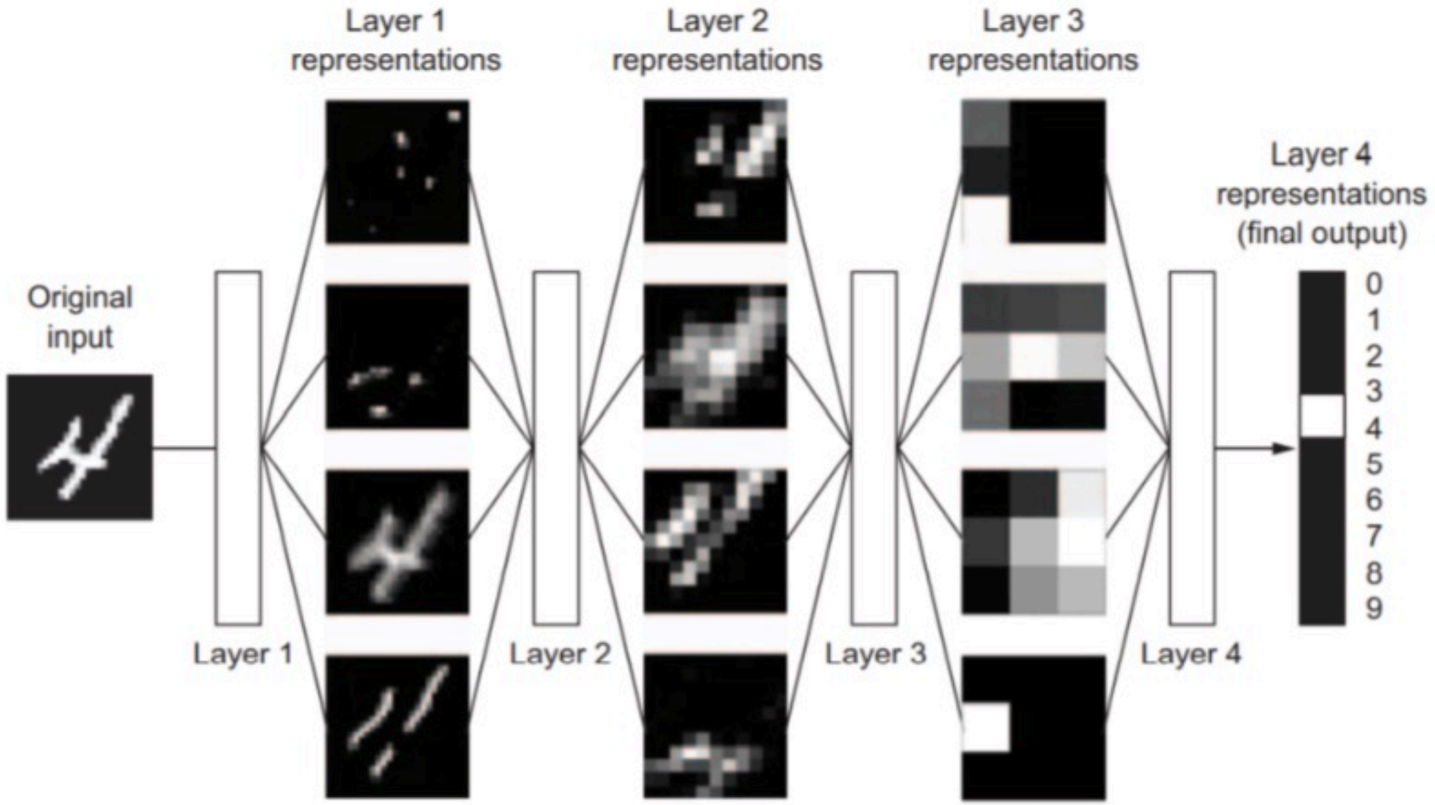
more data +
bigger models +
more computation

(Better algorithms, new insights and improved techniques always help, too!)

Results Get Better With More Data, Larger Models, More Compute
Slide by Jeff Dean, All Rights Reserved.

What is Deep Learning?

Feature Learning



*…a kind of learning where the representation you form have **several levels of abstraction**, rather than a direct input to output*

What is Deep Learning?

The reasons for boost in Deep Learning are:

- Large amount of training data
- Faster machines and multicore **CPU/GPU**s
- New models, algorithms, ideas;
  - Better, more flexible **learning** of intermediate **representations**
  - Effective **learning methods** for using contexts and **transferring** of learned knowledge between tasks
  - Better **regularization** and **optimization** techniques

What is Deep Learning?

Modern state-of-the-art deep learning is focused on ***training deep (many layered) neural network models***

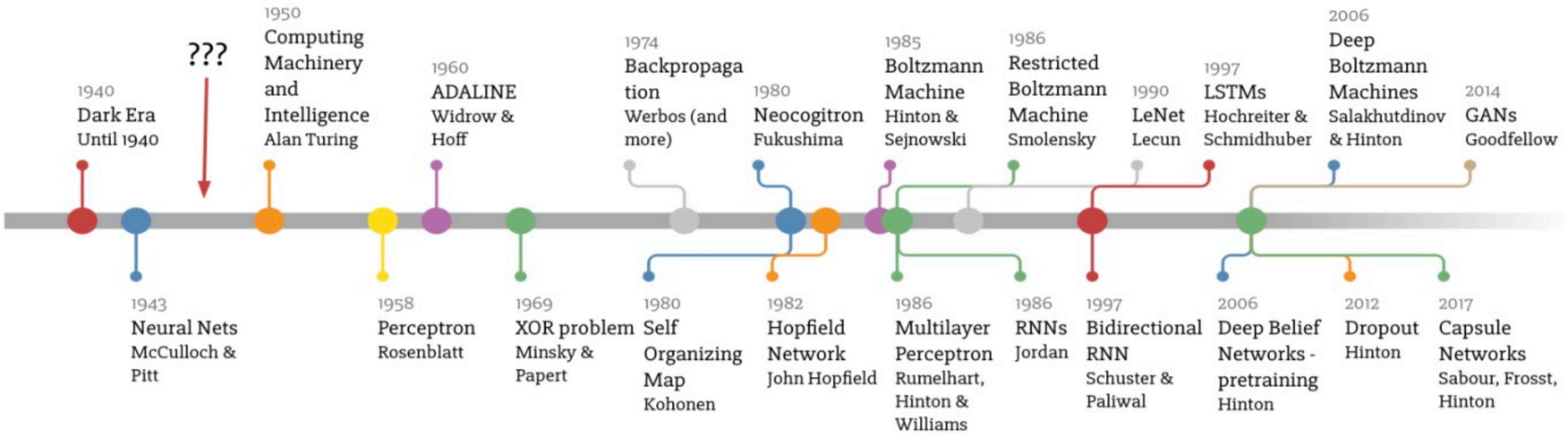
The most popular techniques are:

- Multilayer Perceptron Networks.
- Convolutional Neural Networks.
- Recurrent Neural Networks.

DEEP

Artificial Neural Networks

What is Deep Learning?



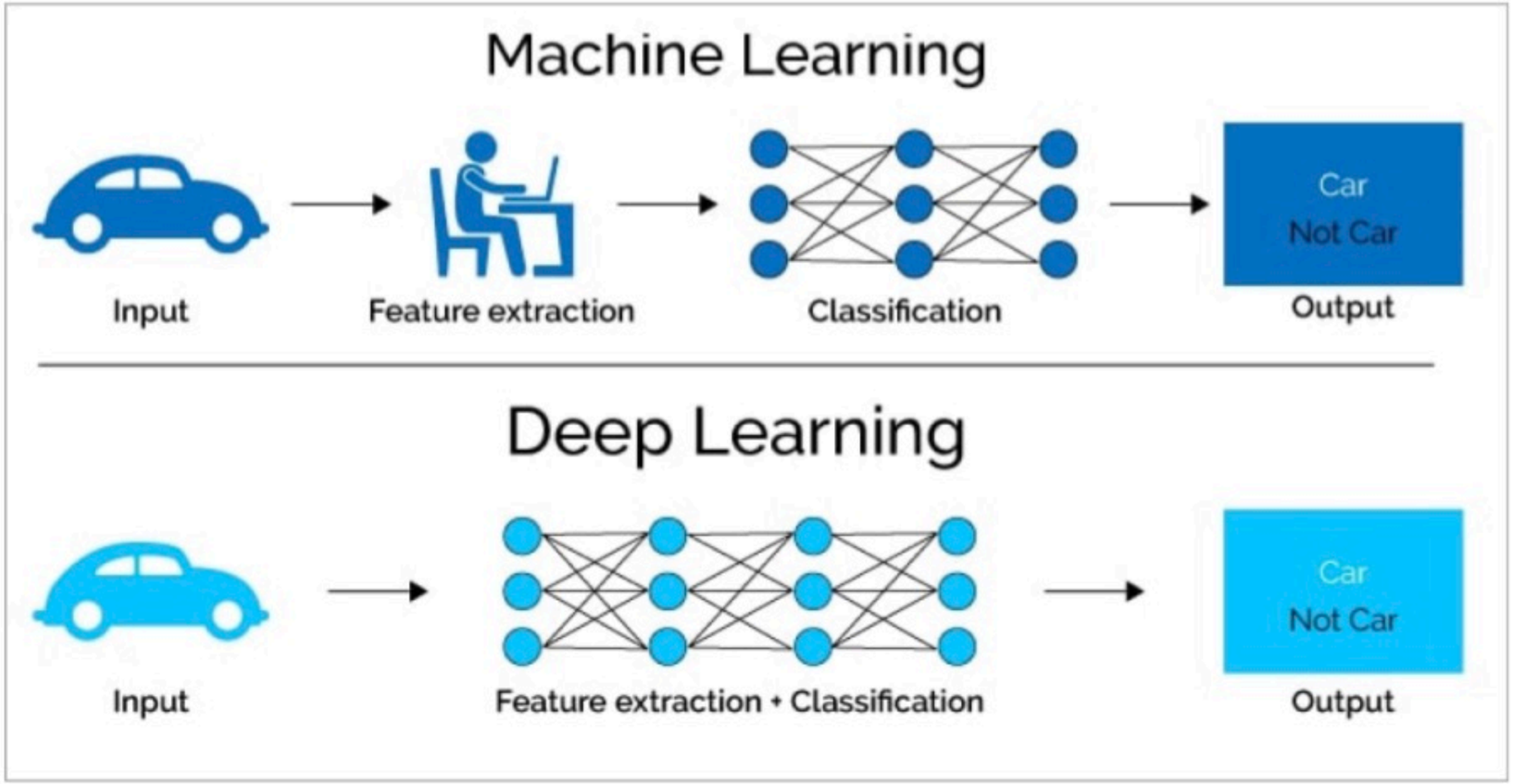
https://www.nytimes.com/2019/03/27/technology/turing-award-ai.html

What is Deep Learning?

Briefly;

- Deep learning is a specialized **subset of machine learning**.
- Deep learning relies on a **layered structure** of algorithms called an artificial neural network.
- Deep learning has **huge data needs** but requires little human intervention to function properly.
- **Transfer learning** is a cure for the needs of large training datasets.

What is Deep Learning?



Deep Learning can process **unstructured data** such as **documents, images, and text**.

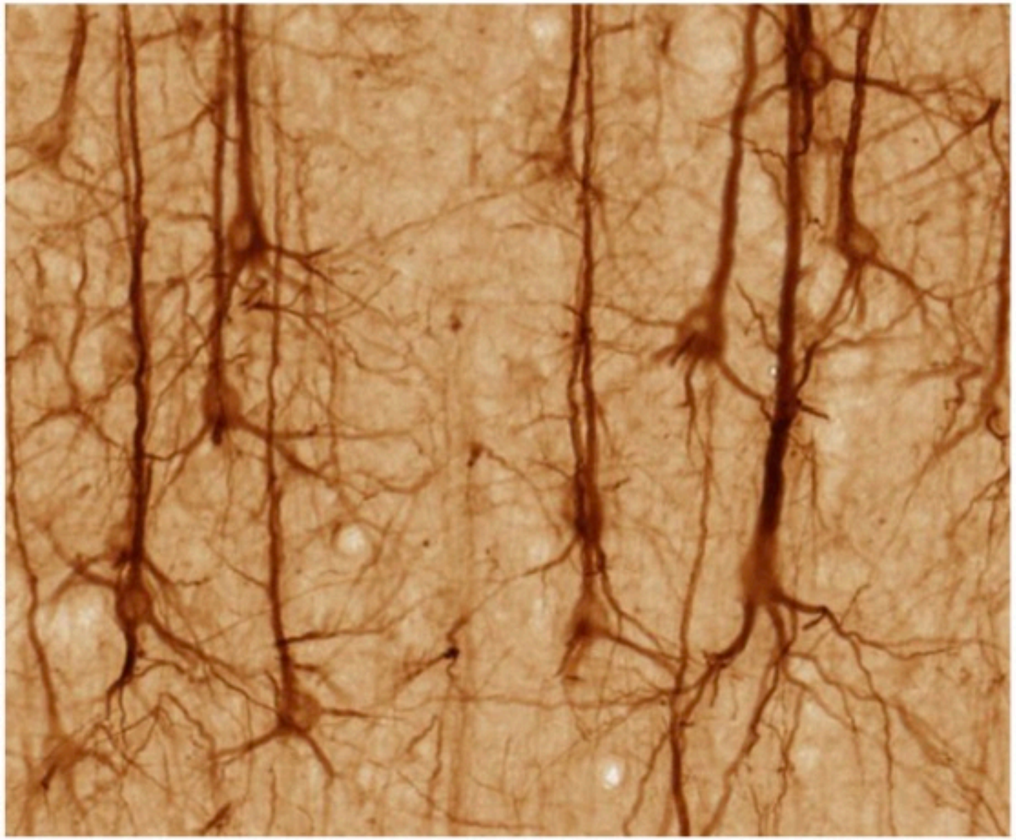
The deep learning algorithm doesn't need a software engineer to identify features but is capable of automatic feature engineering through its neural network. (Source: softwaretestinghelp.com)

Perceptron Model & Neural Networks

Perceptron Models

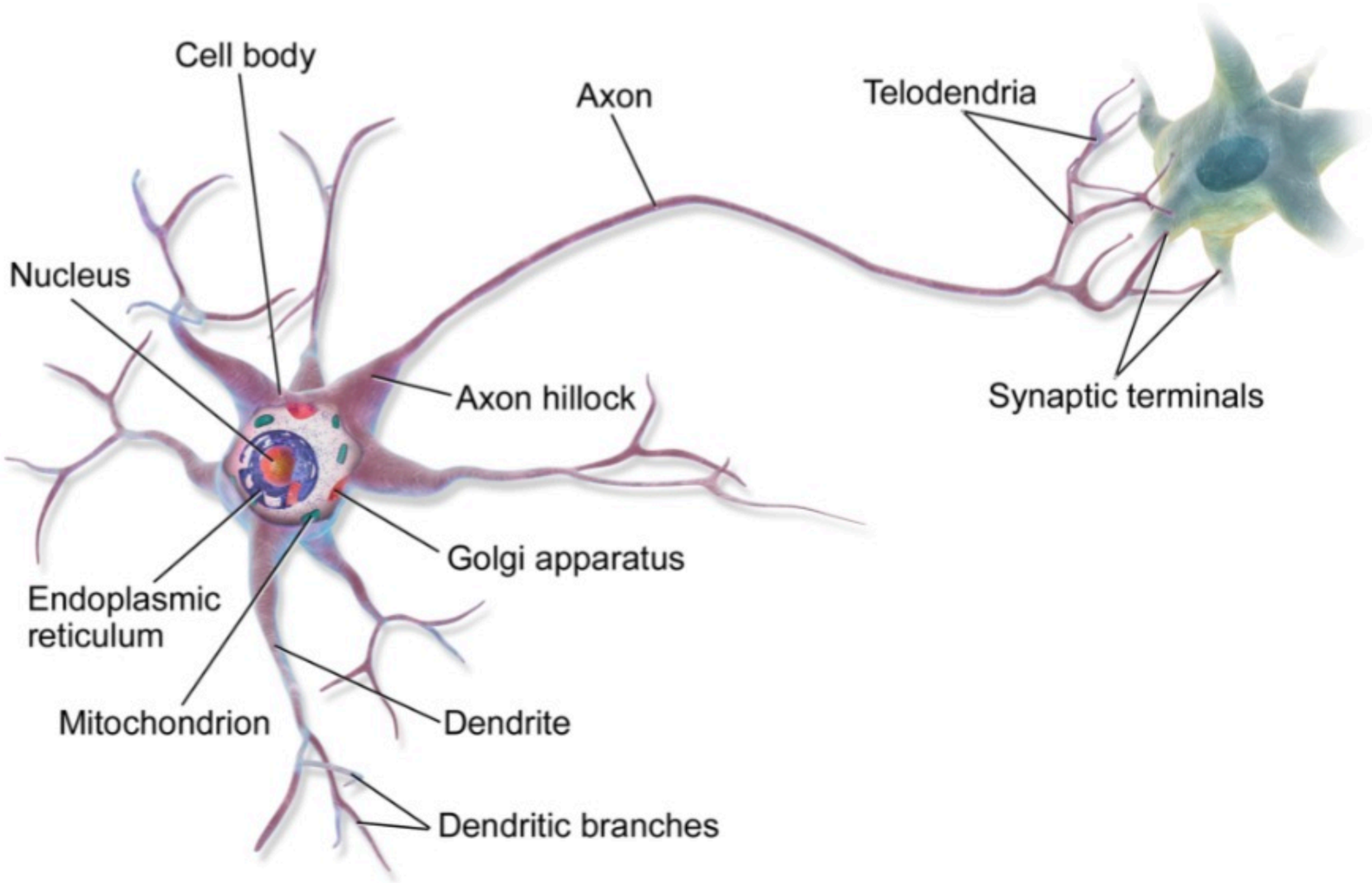
Single Biological Neuron

Neurons in the Cortex



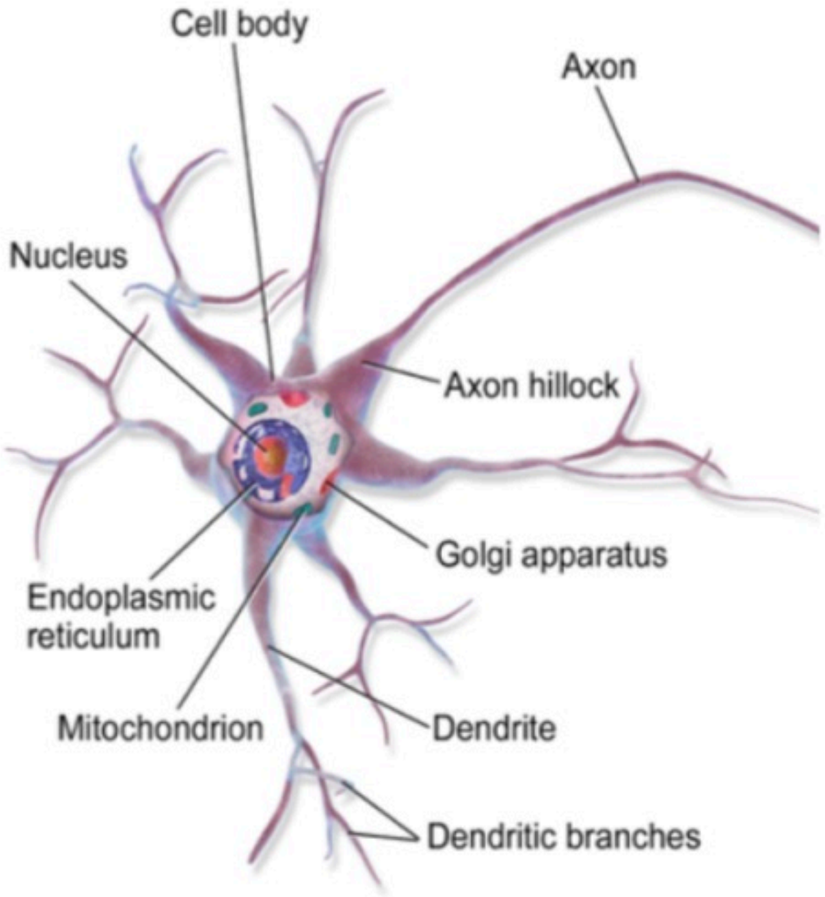
Single Biological Neuron

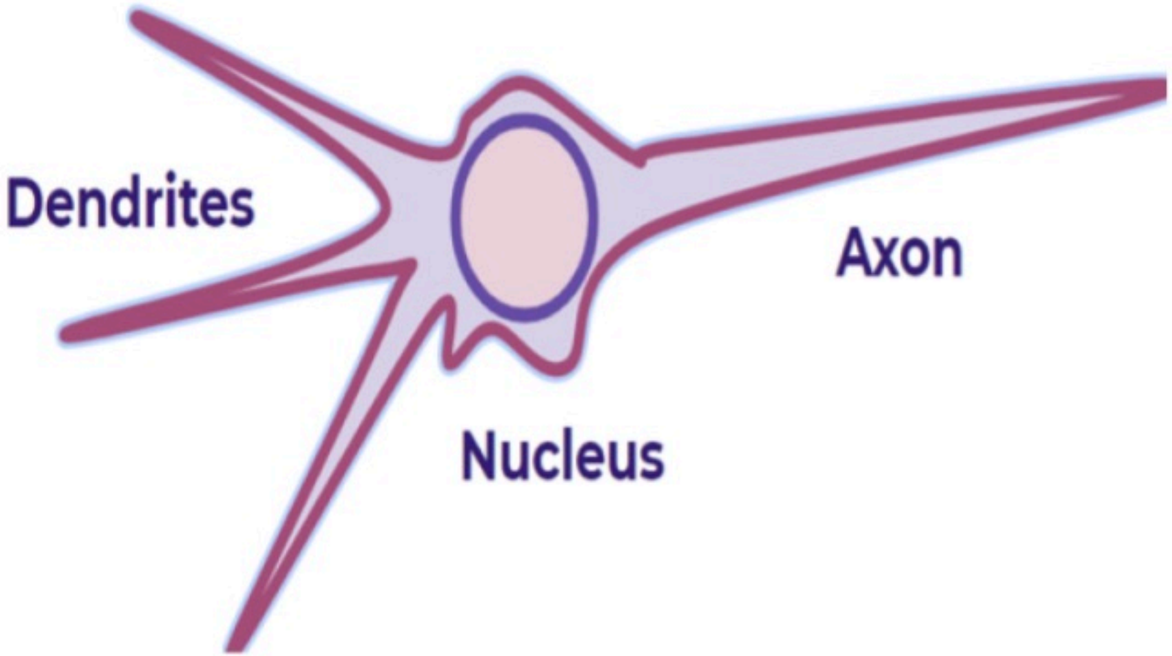
A Single Biological Neuron



Perceptron Models

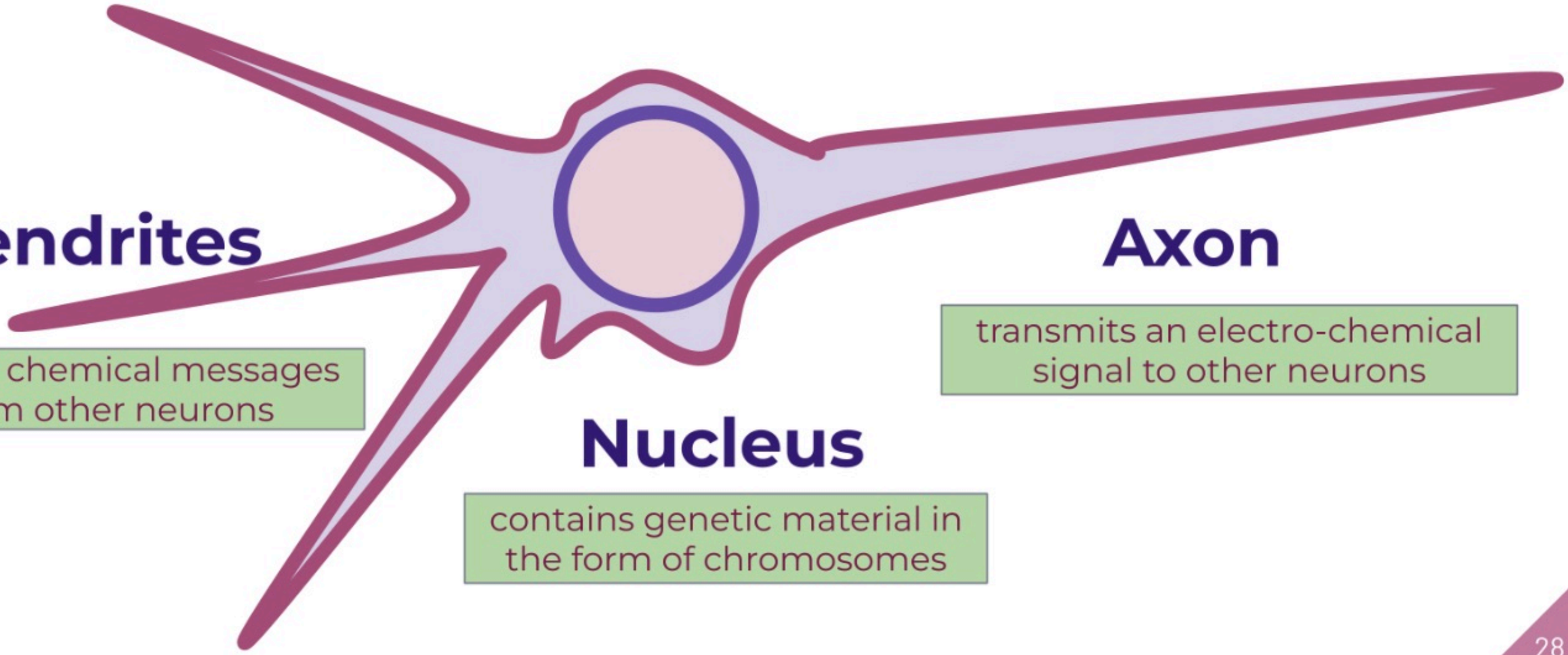
Simplifying the Single Biological Neuron





Perceptron Models

Simplified Biological Neuron Model



Dendrites

receive chemical messages from other neurons

Axon

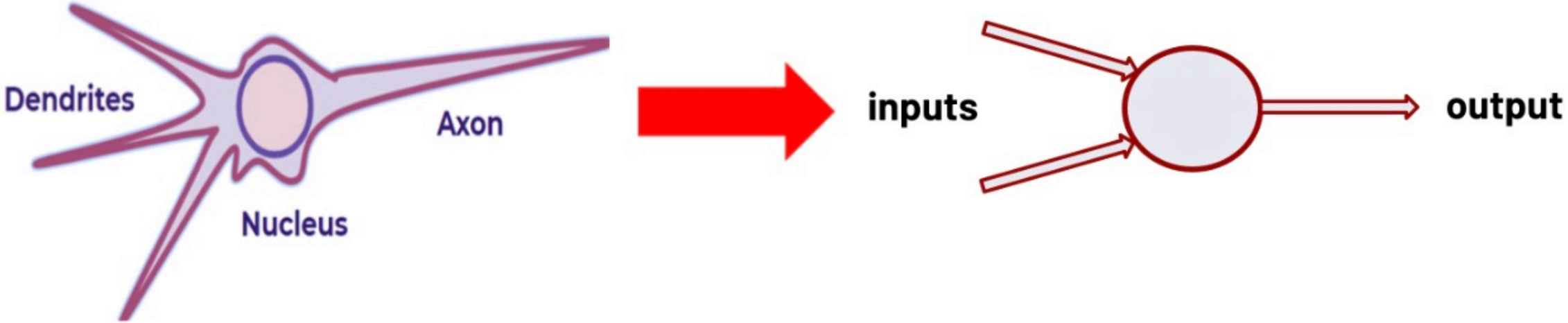
transmits an electro-chemical signal to other neurons

Nucleus

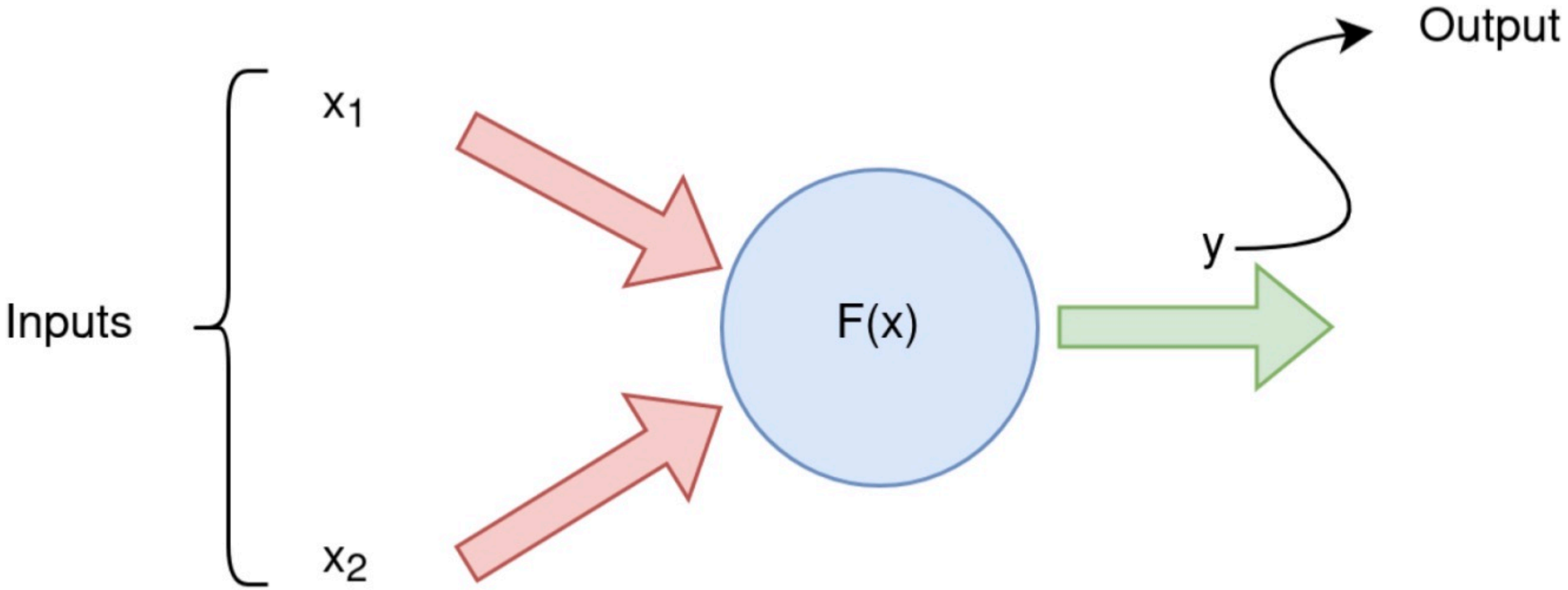
contains genetic material in the form of chromosomes

Perceptron Models

Converting the Simplified Single Biological Neuron to Perceptron Model

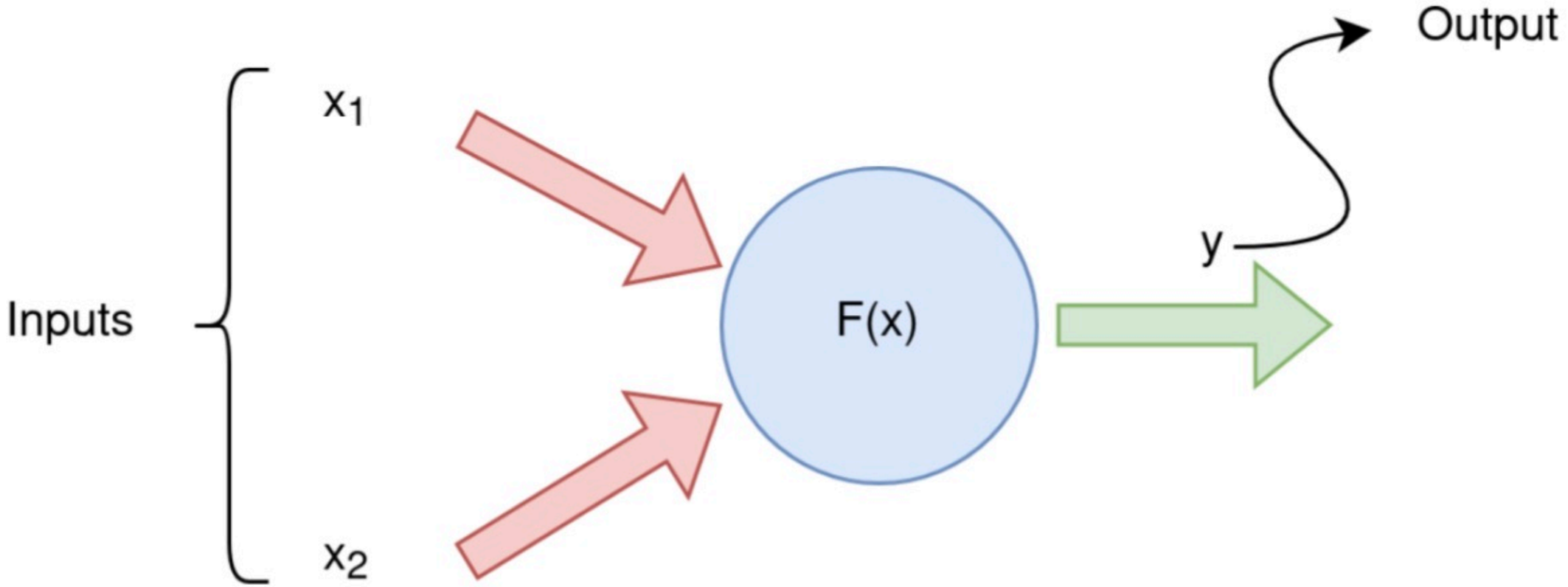


Perceptron Models



Perceptron Models

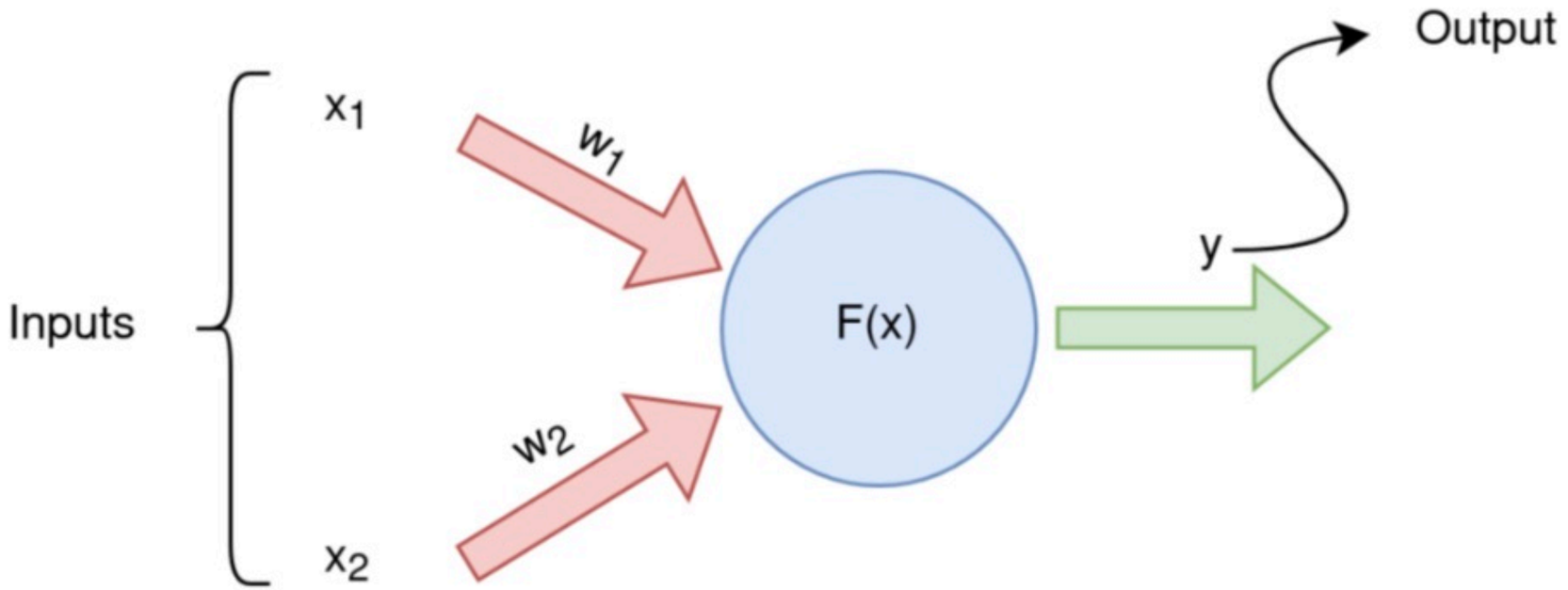
If it is a sum function, then; $y = x_1 + x_2$



Perceptron Models

How to insert another parameter in order to learn: w

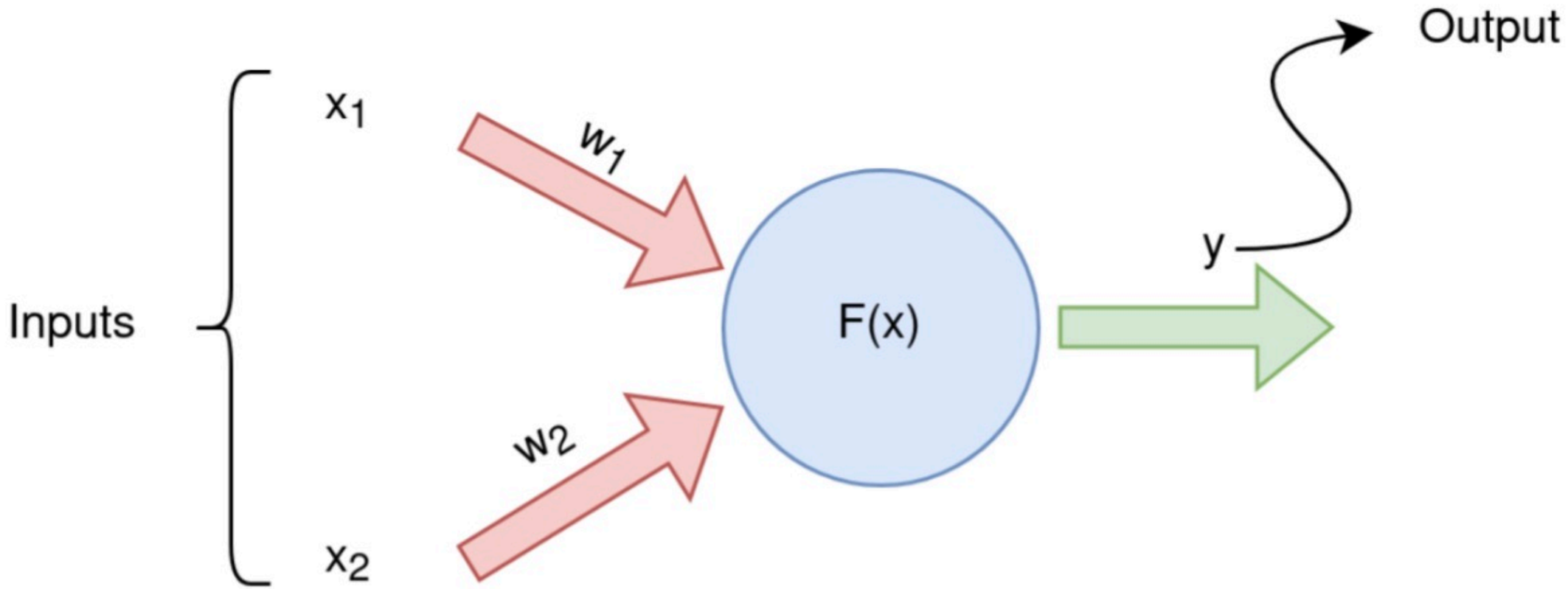
$$y = x_1 \cdot w_1 + x_2 \cdot w_2$$



Now we can update the **weights** to affect **y** (learning)

Perceptron Models

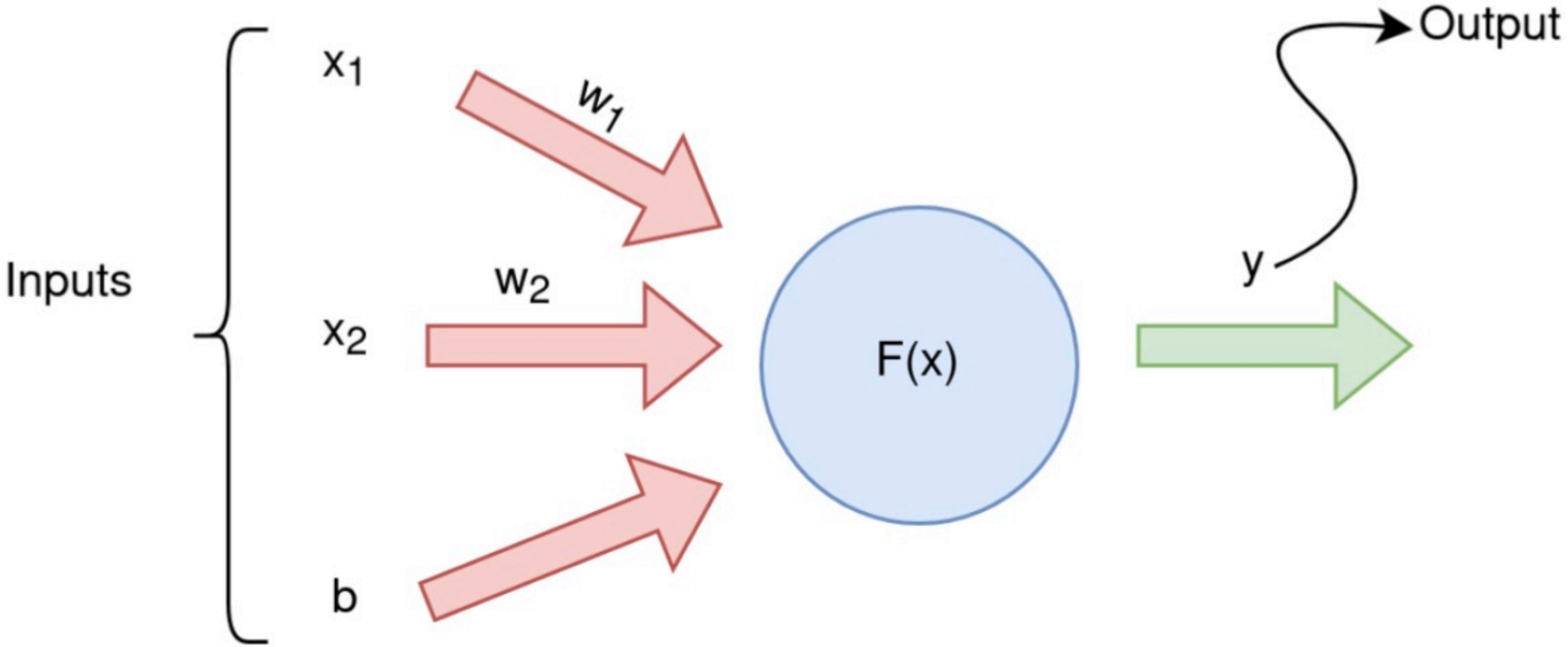
What if an **x** is zero? **w** can not change anything by itself



Perceptron Models

We should add a **bias** term **b** to the inputs.

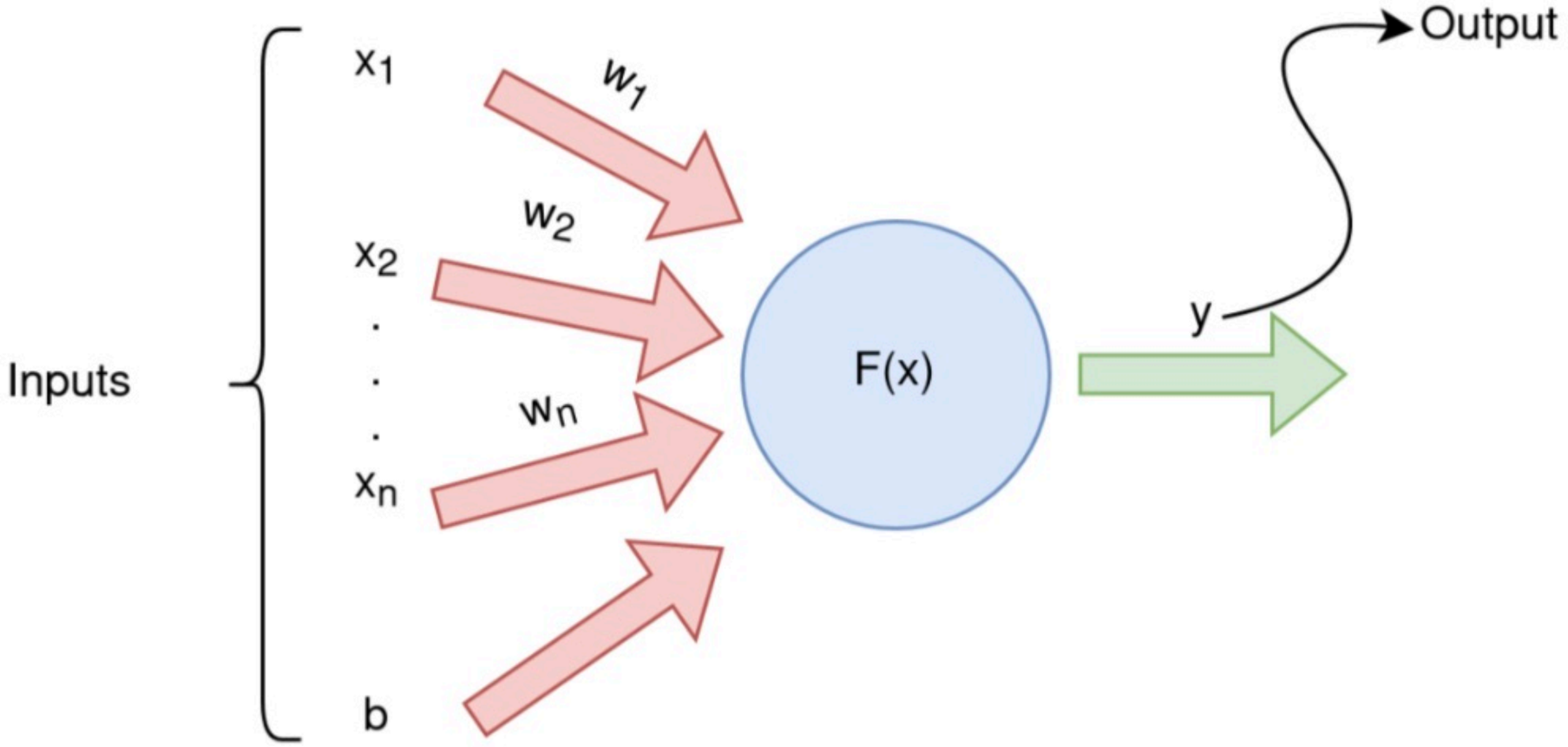
Now; $y = x_1 \cdot w_1 + x_2 \cdot w_2 + b$



Perceptron Models

Generalization of the formula:

$$y = x_1 \cdot w_1 + x_2 \cdot w_2 + \ldots + x_n \cdot w_n + b$$



Perceptron Models

input x weight + bias = output

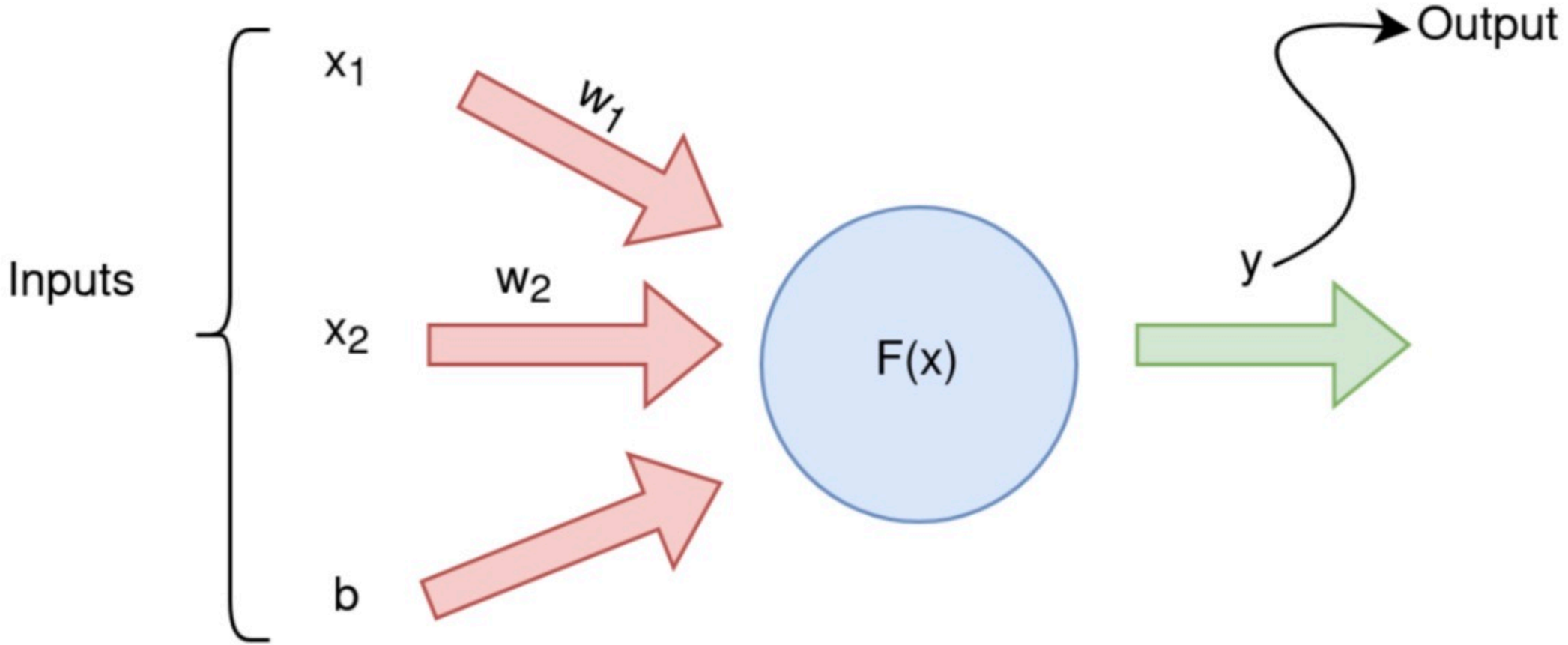
The formula is: $z = \sum x_i \cdot w_i + b$ b: *bias*

$$z = x_1 \cdot w_1 + x_2 \cdot w_2 + x_3 \cdot w_3 + \cdots + x_n \cdot w_n + b$$

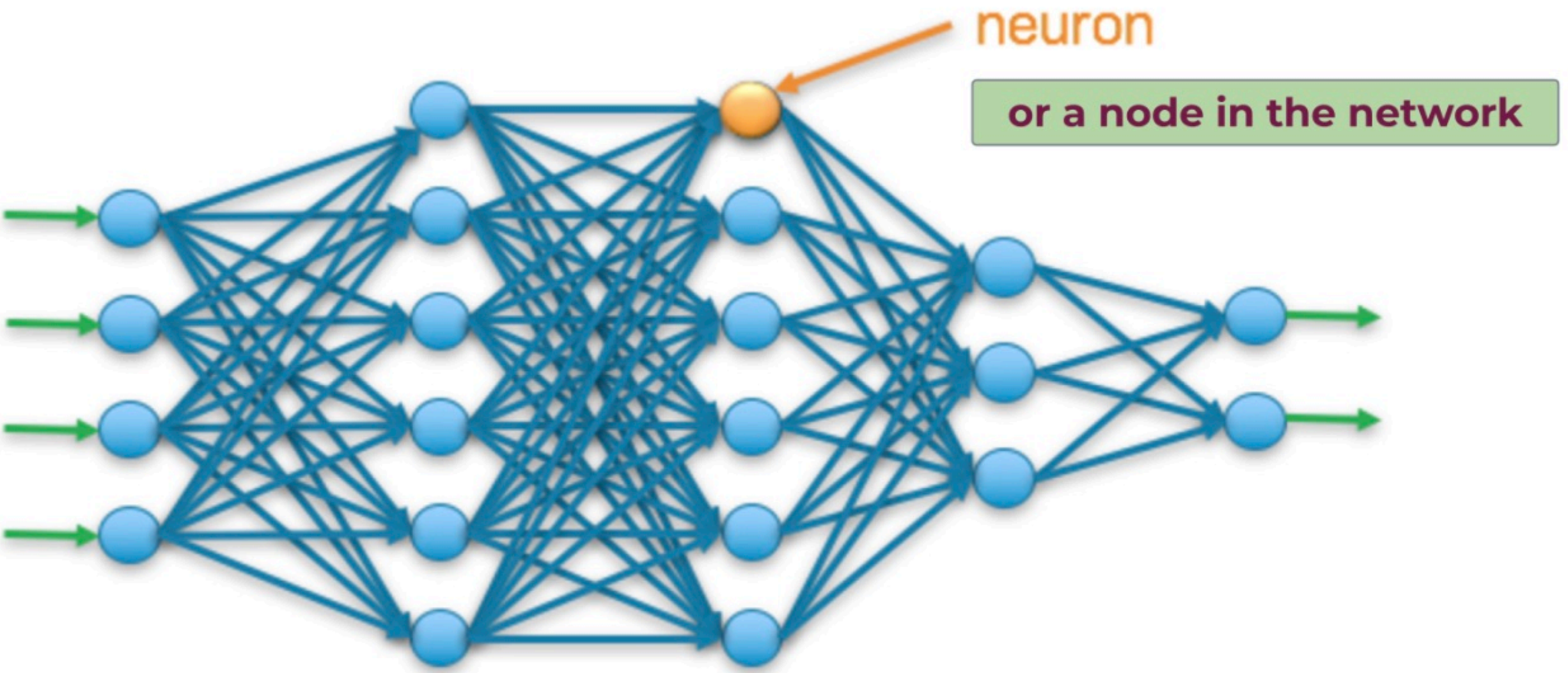
Neural Networks

What is a Neuron ?

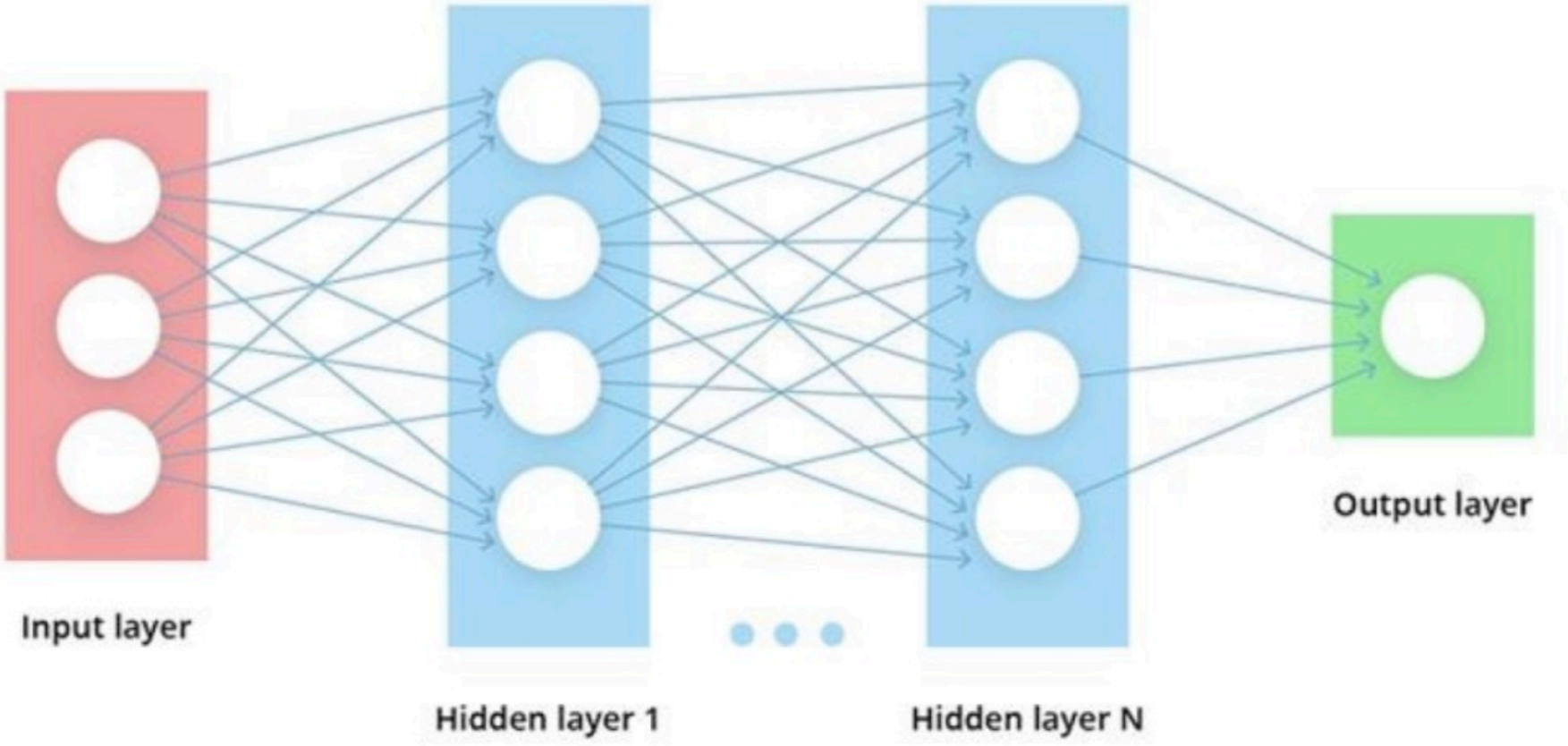
A single perceptron model = **A neuron** = **A node in a network**



Neurons in a Network



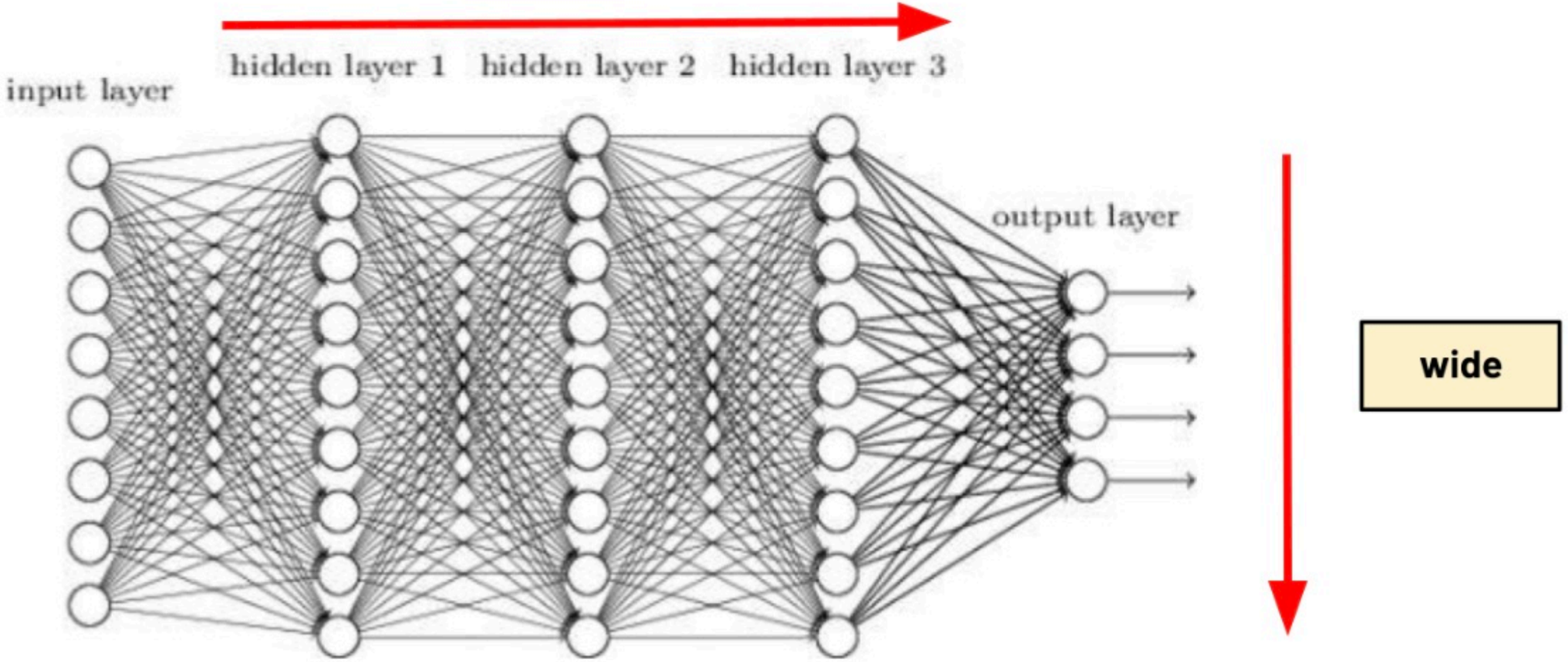
Layers



Layers

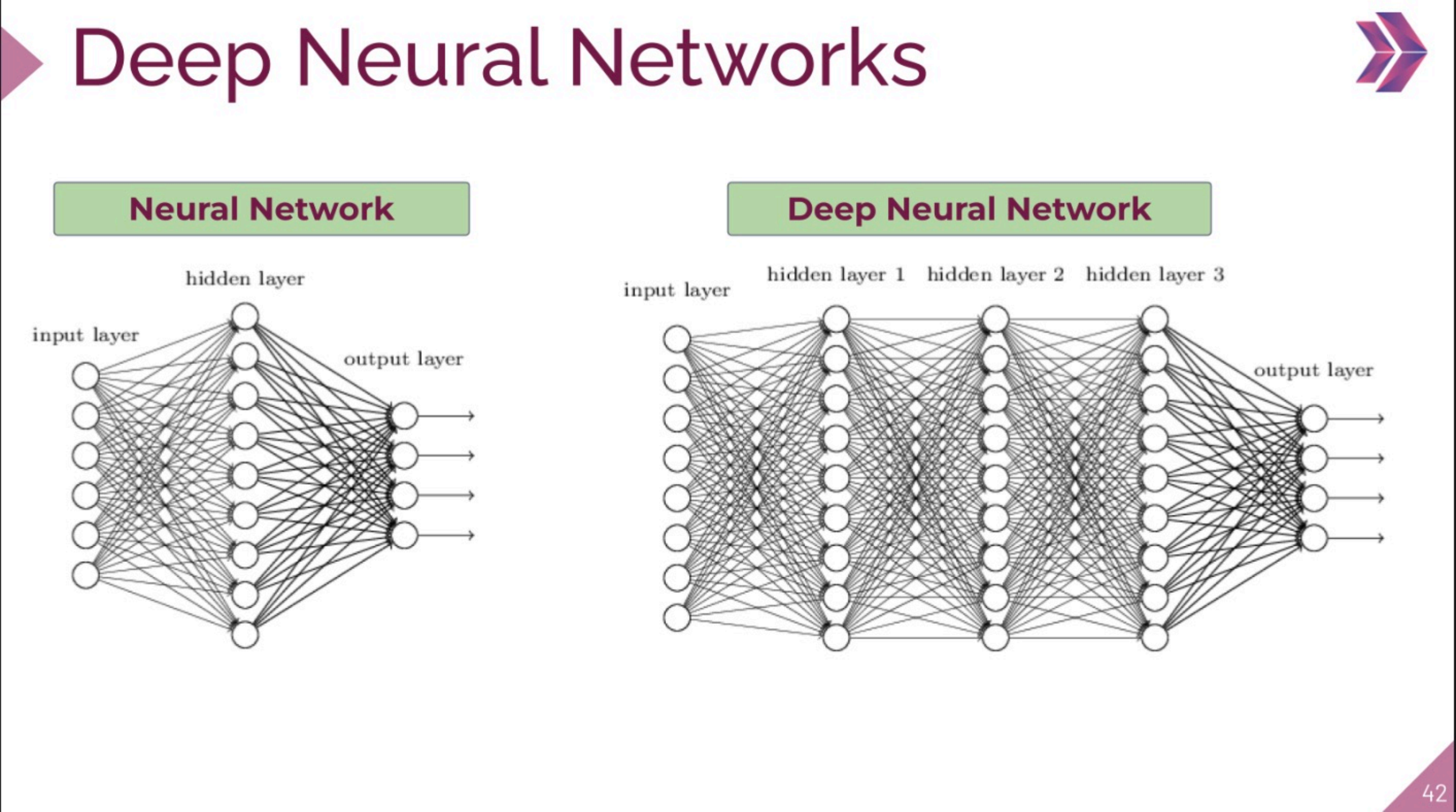
Fully Connected Layers → **Fully Connected Neural Network**

deep



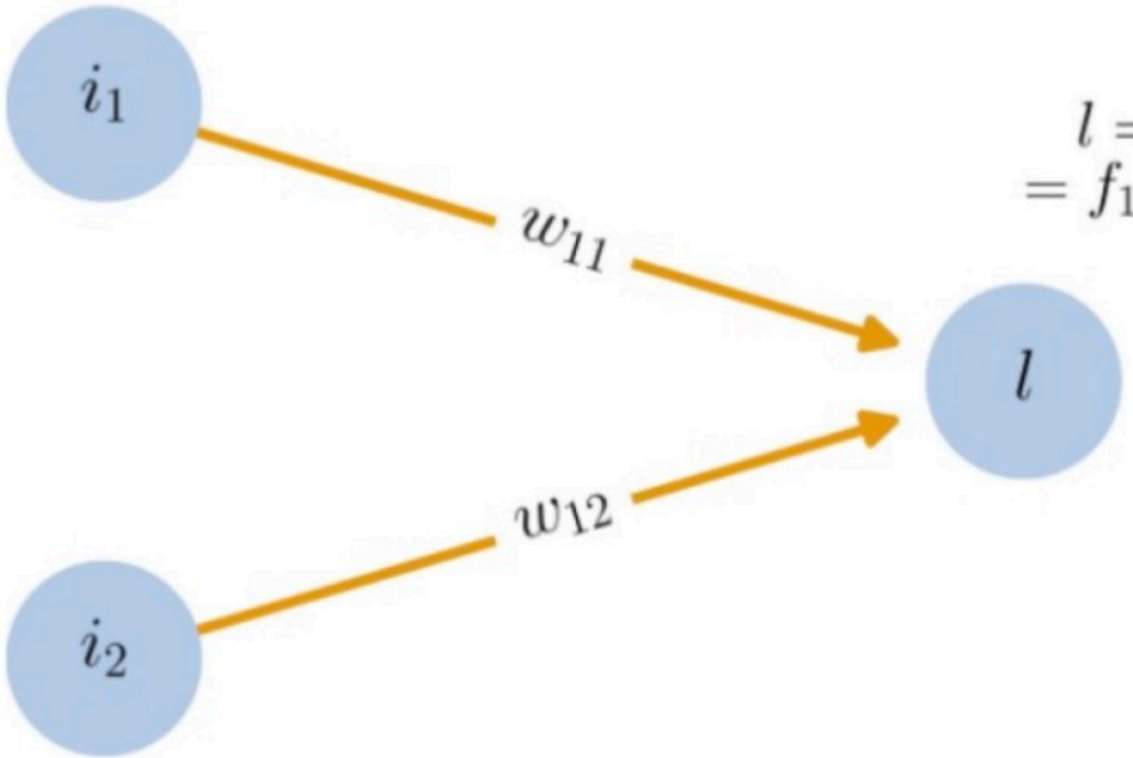
wide

Deep Neural Networks



Deep Learning vs Linear Regression

Linear Regression



$$l = w_{11} \times i_1 + w_{12} \times i_2 = f_1(i_1, i_2)$$

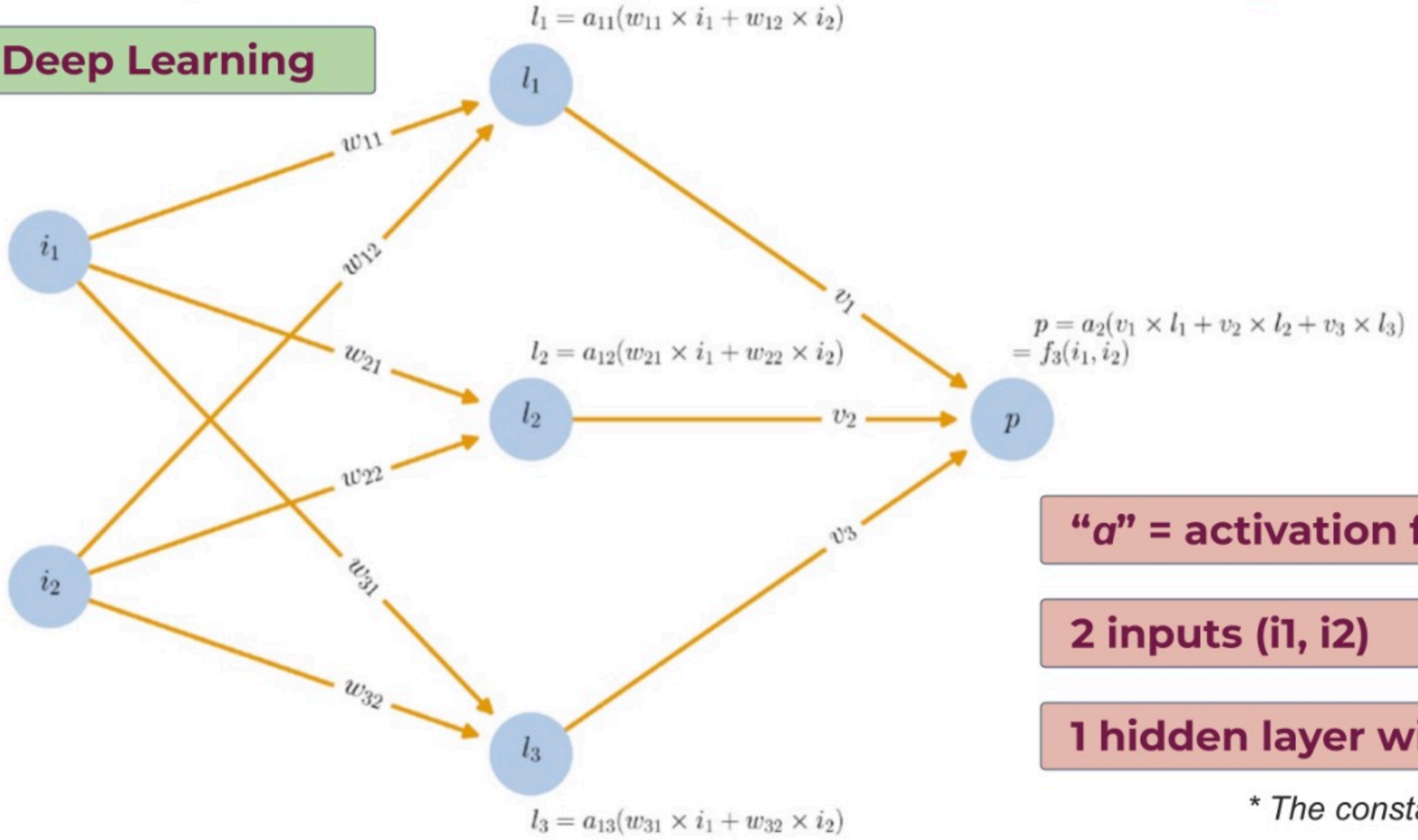
prediction of a house price with two inputs

The actual formula is:
*f1(i1, i2) = w11 * i1 + w12 * i2 + **c**.*
But the constant (c) is ignored in the picture

https://towardsdatascience.com/a-gentle-journey-from-linear-regression-to-neural-networks-68881590760e

Deep Learning vs Linear Regression

Deep Learning



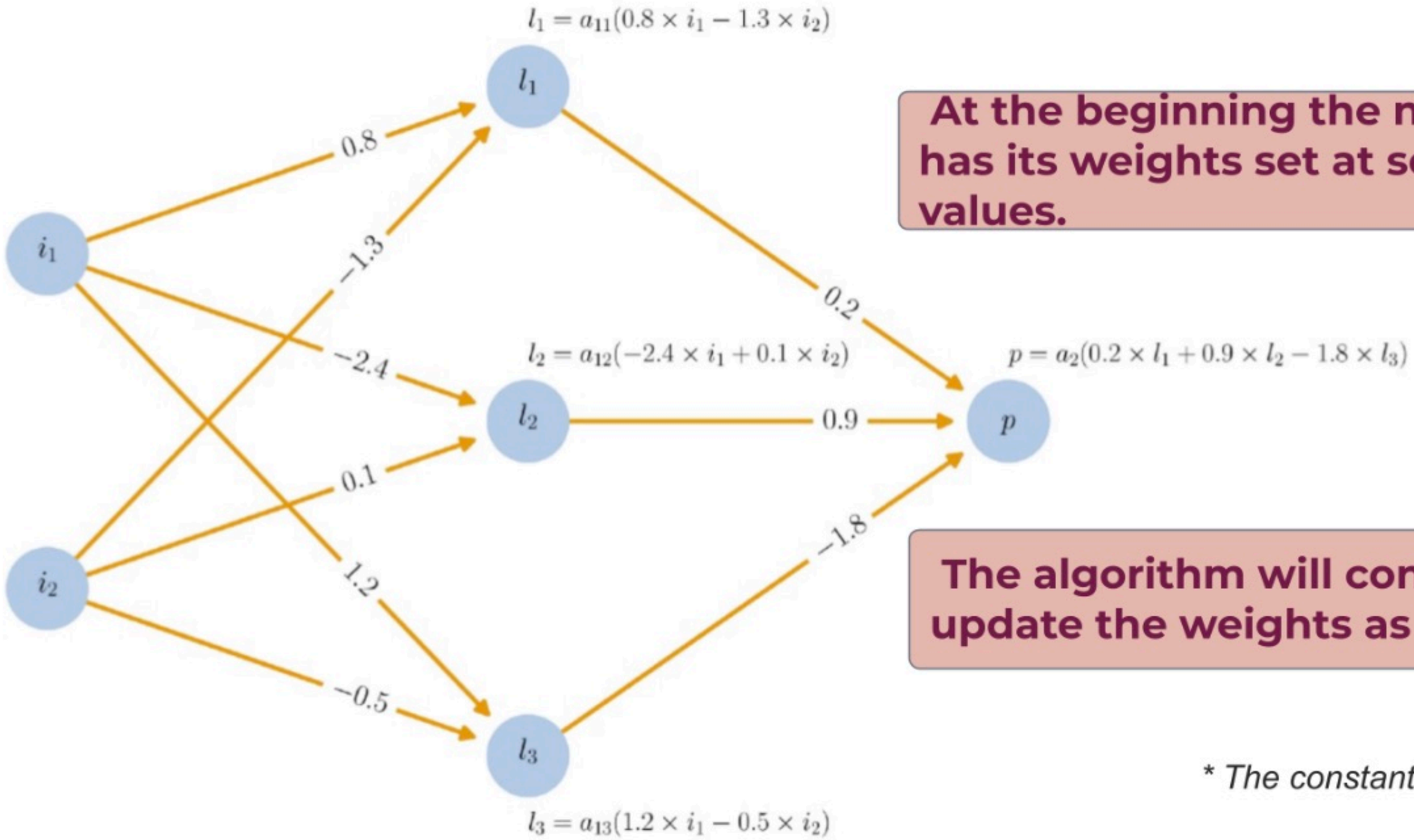
"*a*" = activation function

2 inputs (i1, i2)

1 hidden layer with 3 neurons

** The constants are ignored.*

Deep Learning vs Linear Regression



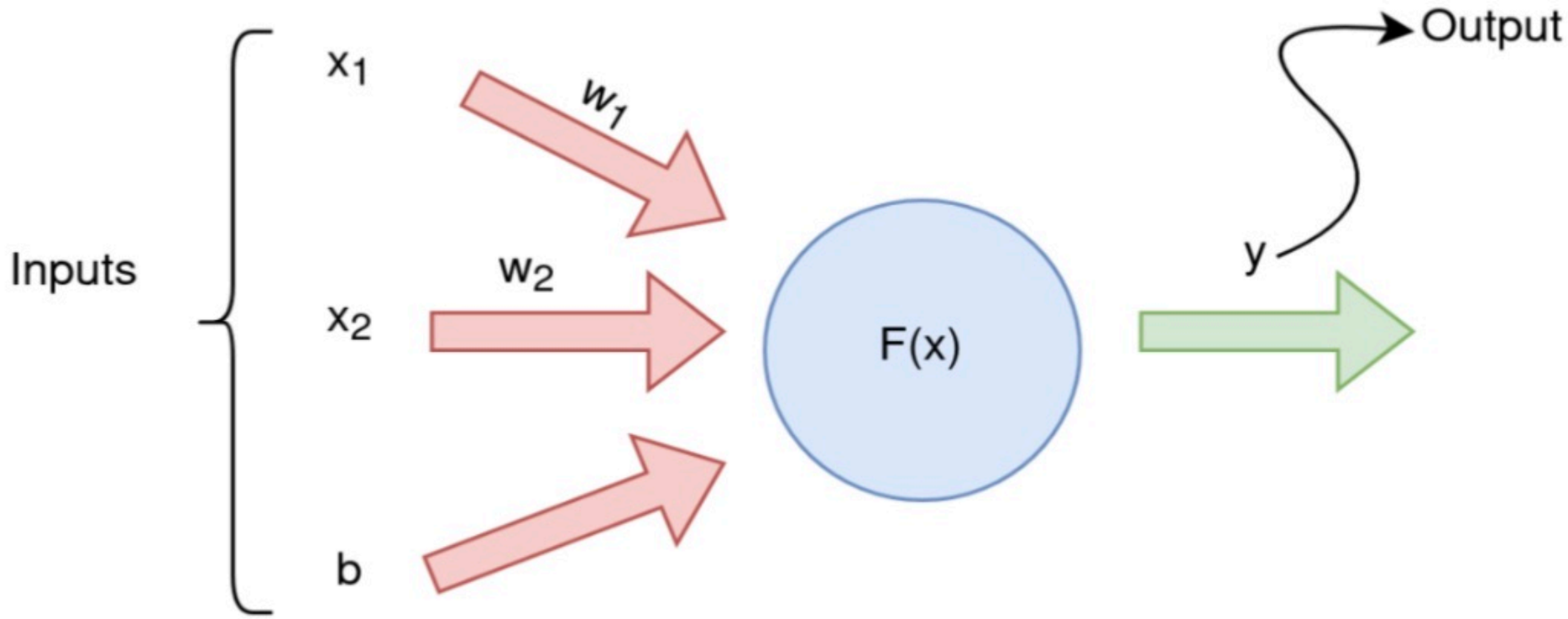
At the beginning the network has its weights set at some values.

The algorithm will continuously update the weights as it learns.

** The constants are ignored.*

Activation Functions

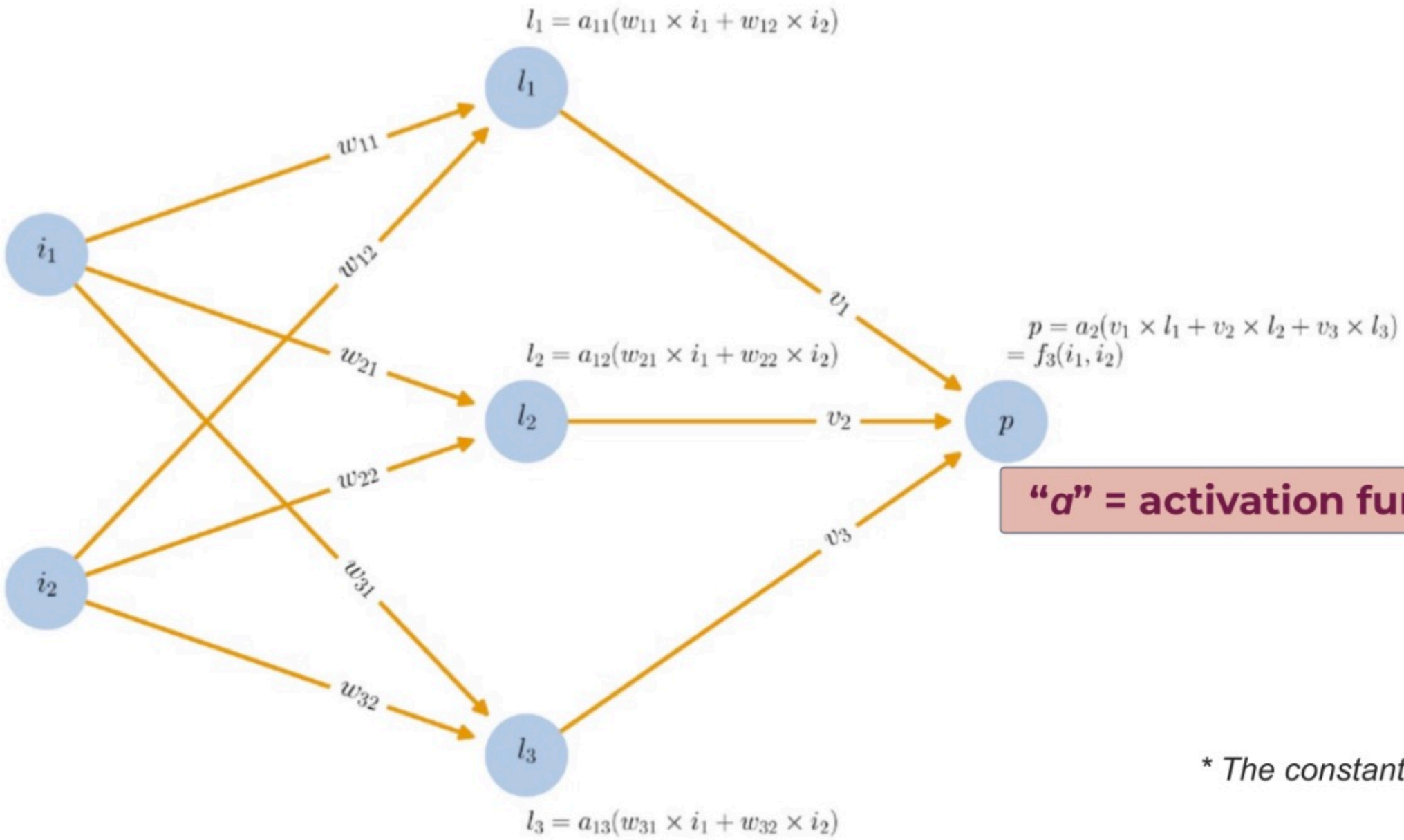
Activation Functions



$$\hat{y} = \sum_{i=1}^{n} x_i w_i + b$$

The value of Y can be anything ranging from -inf to +inf. The neuron really doesn't know the bounds of the value.

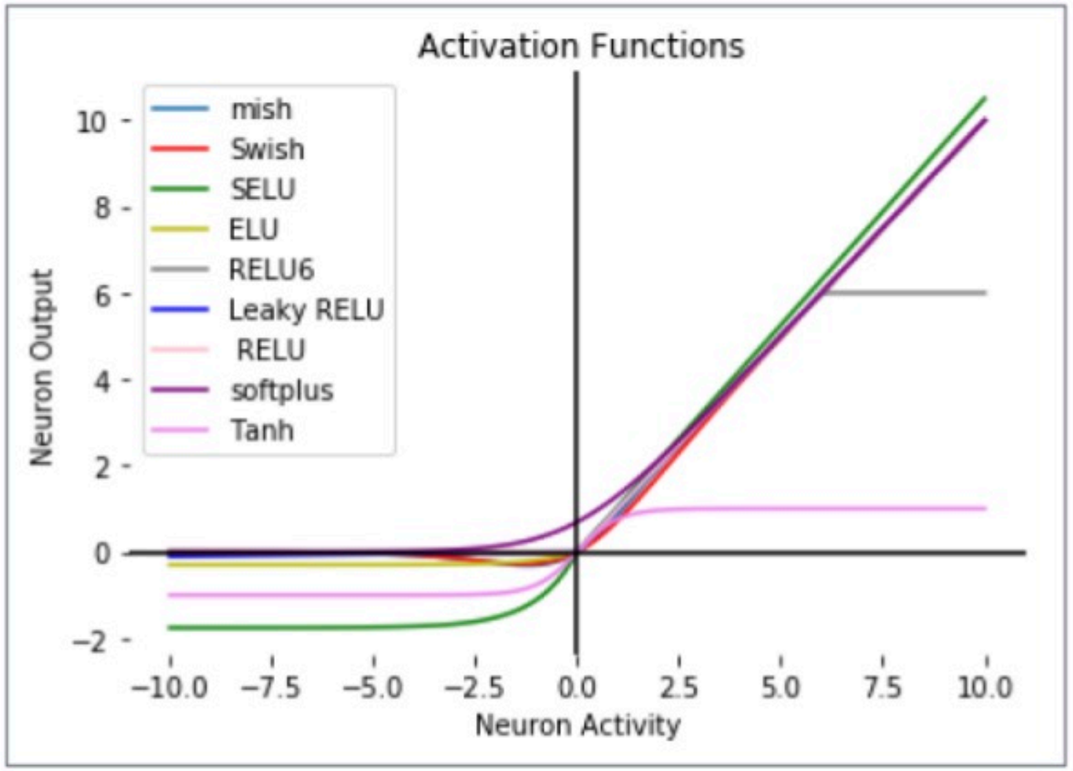
Activation Functions



"a" = activation function

** The constants are ignored.*

Activation Functions



- The activation function of a node **defines the output** of that node given an input or set of inputs.
- Activation functions are **decision making units** of neural networks.

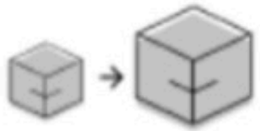
Activation Functions

Role of the Activation function

- Activation functions decide if a neuron will be **fired or not** and **strength of the signal** outputted by the neuron.
- Activation functions determine the deep learning model's **accuracy and the computational efficiency** for training a model.
- Activation functions have to be computationally very efficient as it decides on the network's **convergence speed** or if the network will **ever converge**.

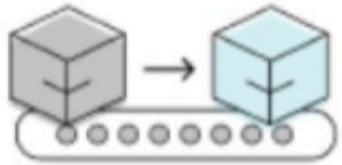
Activation Functions

How it works?



Take input and multiply by the neuron's weight.

Add bias*



Feed the result, x, to the activation function: f(x)

Take the output and transmit to the next layer of neurons.

Activation Functions

Type of Activation Functions

Binary Step Function 1

Linear Activation Function 2

Non-Linear Activation Function 3

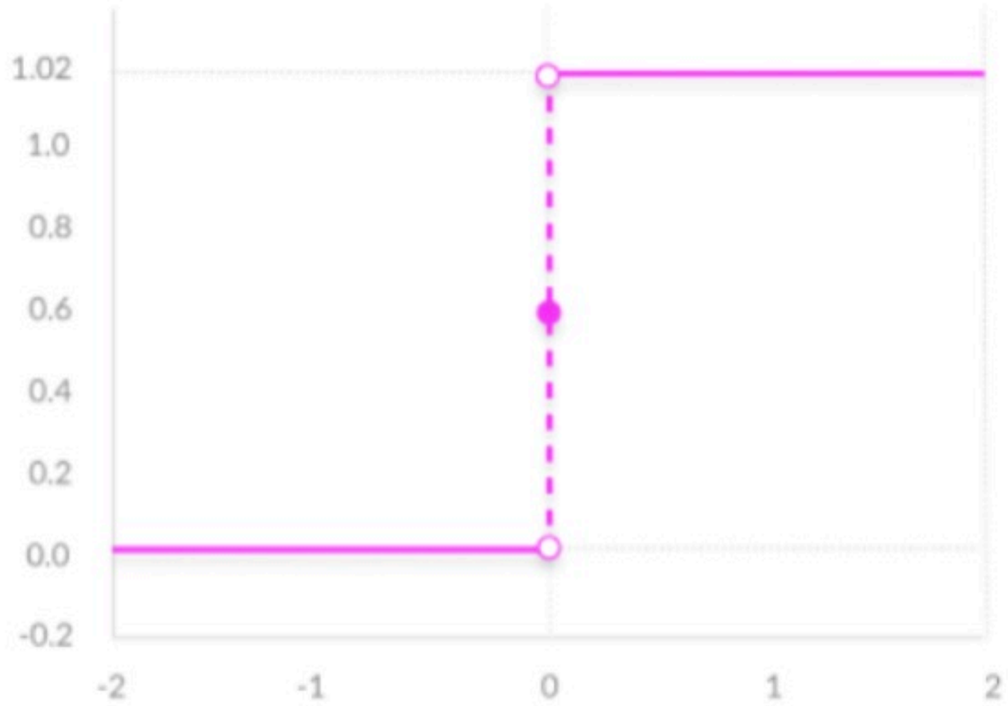
- **Sigmoid**
- **Softmax**
- **ReLU**
- **TanH**
- **Leaky ReLU**

** Modern neural network models use non-linear activation functions*

Type of Activation Functions

Binary Step Function

No multi classification



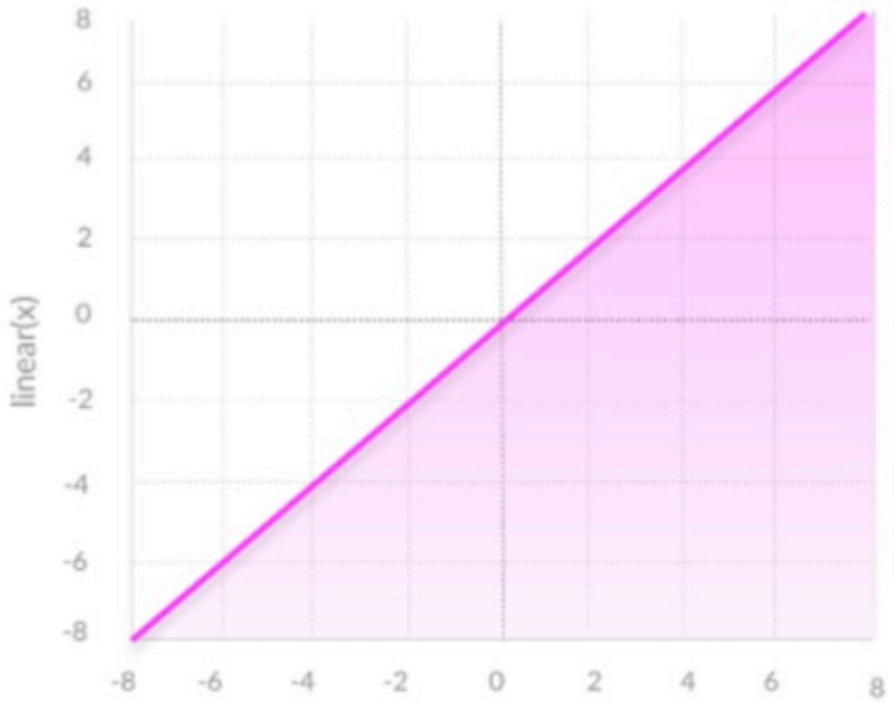
The function produces **binary output**. The function produces **1 (or true)** when input **passes threshold limit** whereas it produces **0 (or false)** when input does **not pass threshold**.

Type of Activation Functions

Linear Activation Function

No backpropagation

One hidden layer



$f(x) = x$

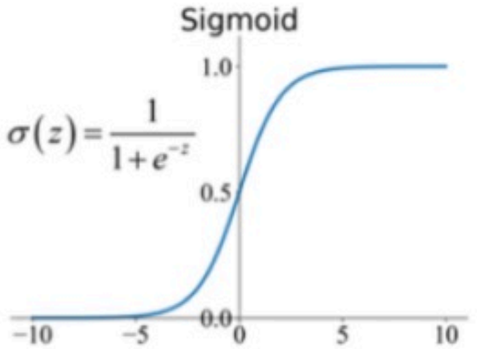
It takes the inputs, multiplied by the weights for each neuron, and creates an output signal proportional to the input.

Type of Activation Functions

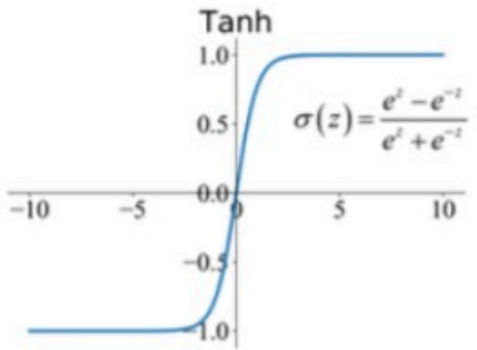
Non-Linear Activation Functions

Allow backpropagation

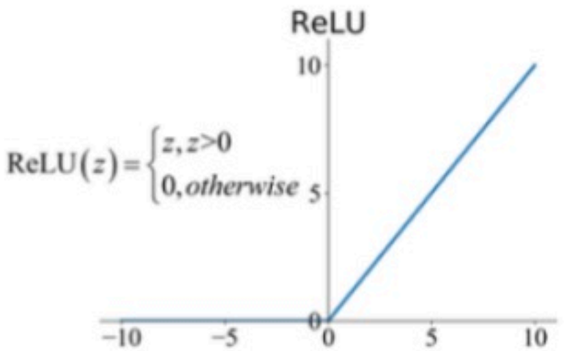
Allow multiple hidden layers



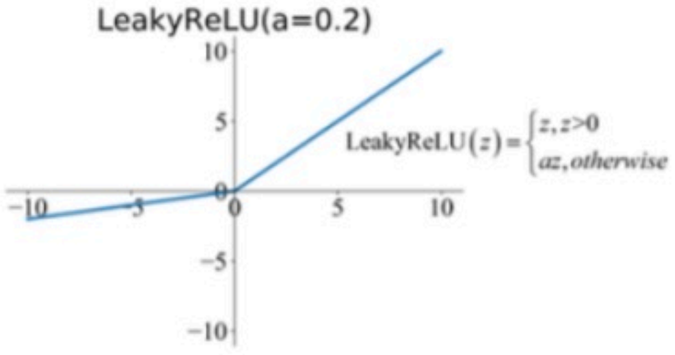
(a)



(b)



(c)



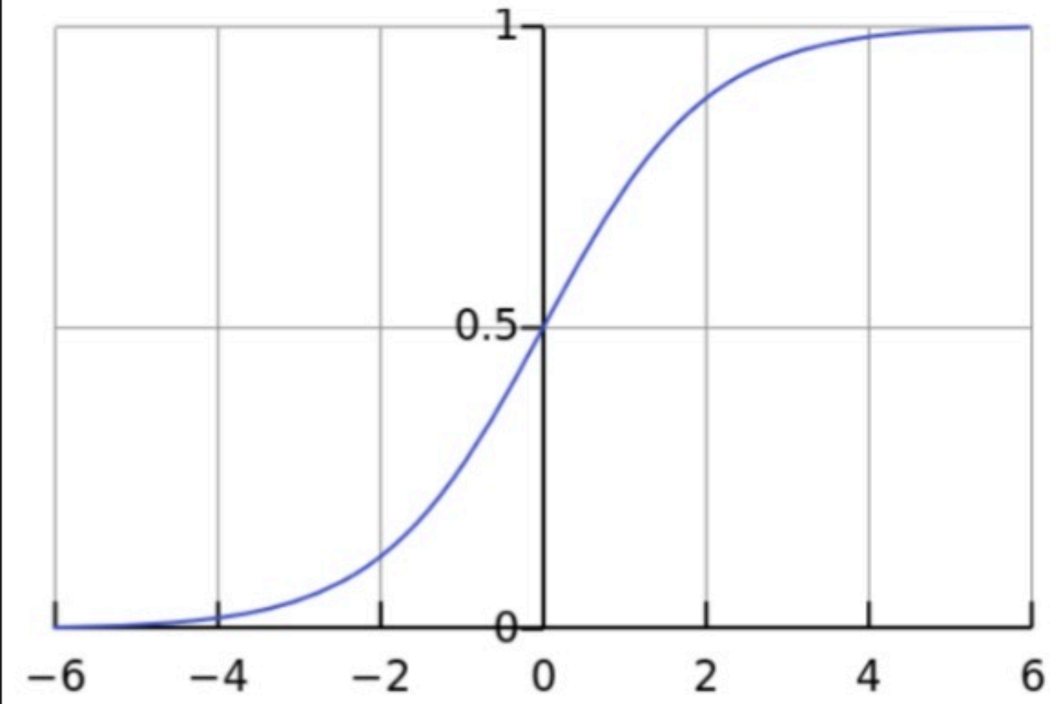
(d)

Modern neural network models use non-linear activation functions

Sigmoid Activation Functions

Output values bound between **0 and 1**, normalizing the output of each neuron.

$$A = \frac{1}{1+e^{-x}}$$



Pros

- **Smooth gradient**, preventing "jumps" in output values.
- **Clear predictions**—For X above 2 or below -2, tends to bring the Y value (the prediction) to the edge of the curve, very close to 1 or 0. This enables clear predictions.

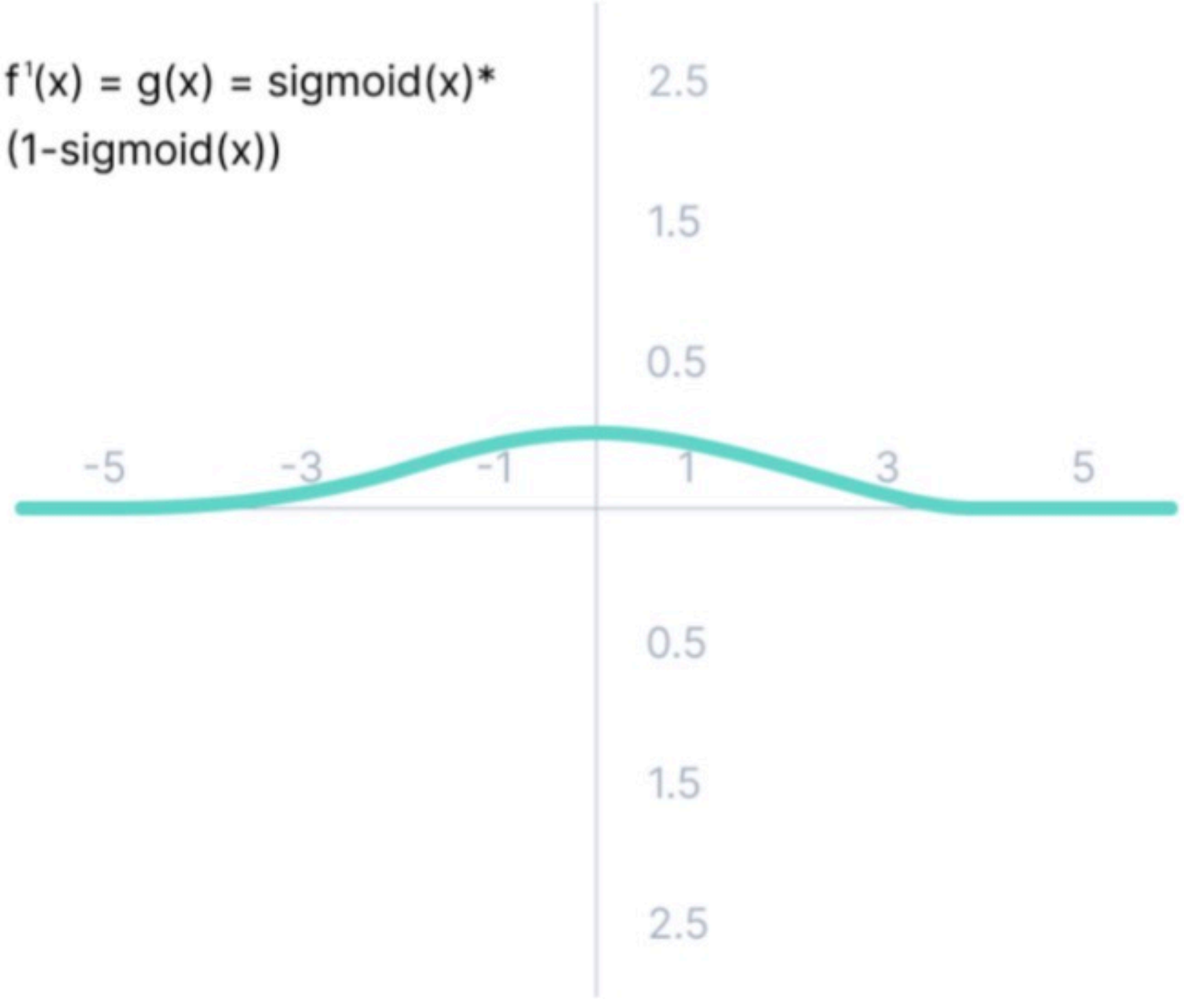
Cons

- **Vanishing gradient**—for very high or very low values of X, there is almost no change to the prediction, causing a vanishing gradient problem. This can result in the network refusing to learn further, or being too slow to reach an accurate prediction.
- **Outputs not zero centered**.
- **Computationally expensive**

Sigmoid Activation Functions

Derivative of the Sigmoid Activation Function

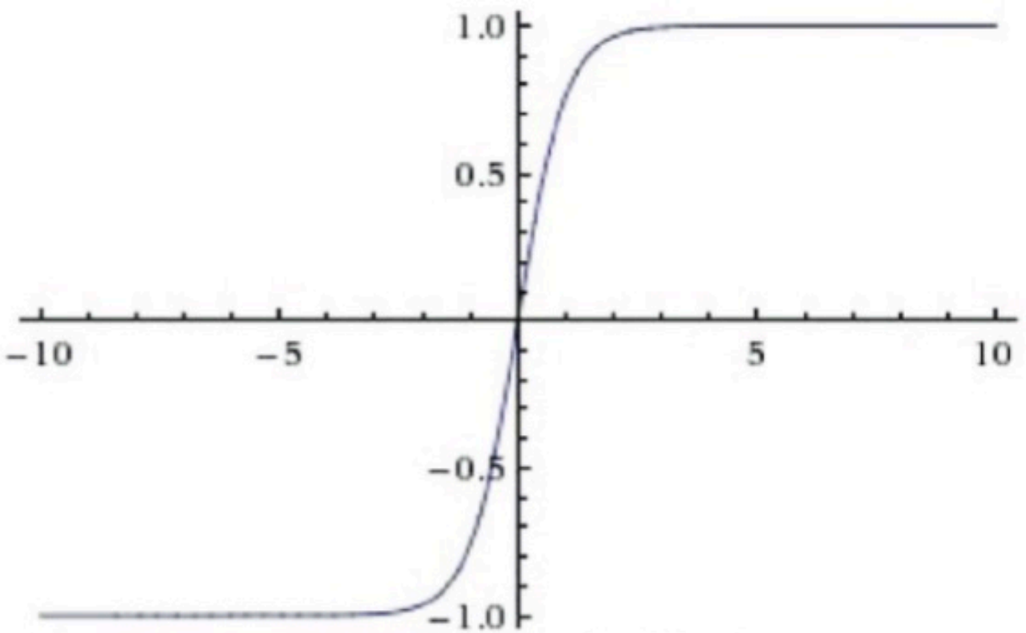
$f^1(x) = g(x) = sigmoid(x)*(1-sigmoid(x))$



TanH Activation Functions

Output values bound between **-1 and 1**.

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



Pros

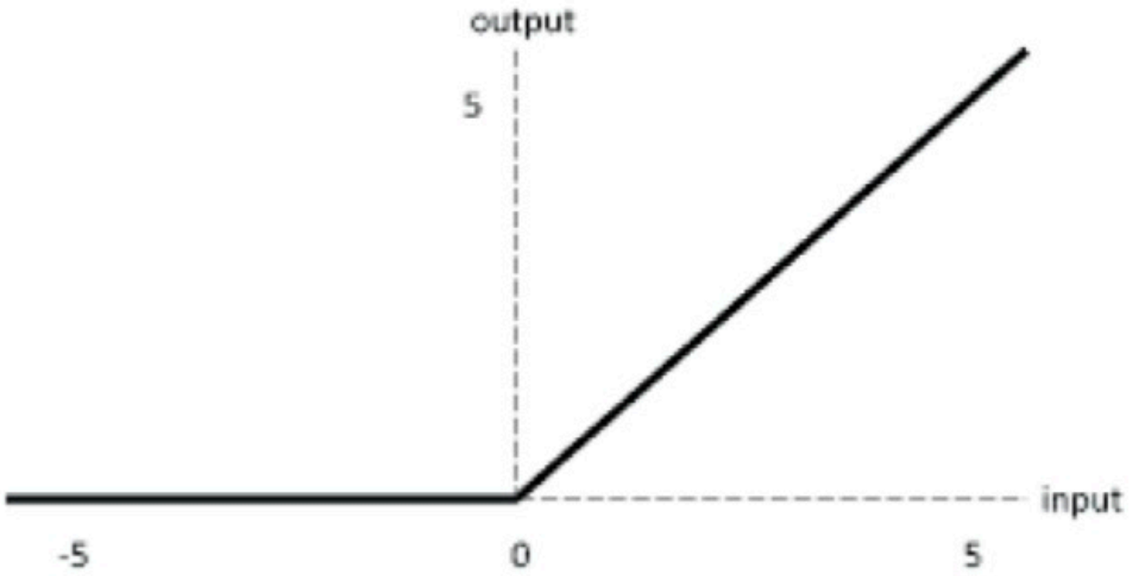
- **Same Advantages with Sigmoid.**
- **More efficient than Sigmoid:** Because it has a wider range for faster learning and grading.
- **Outputs zero centered.**

Cons

- **Vanishing gradient**—for very high or very low values of X, there is almost no change to the prediction, causing a vanishing gradient problem. This can result in the network refusing to learn further, or being too slow to reach an accurate prediction.
- **Computationally expensive**

ReLu Activation Functions

$$f(x) = \begin{cases} 0 & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases} = \max\{0, x\}$$



Pros

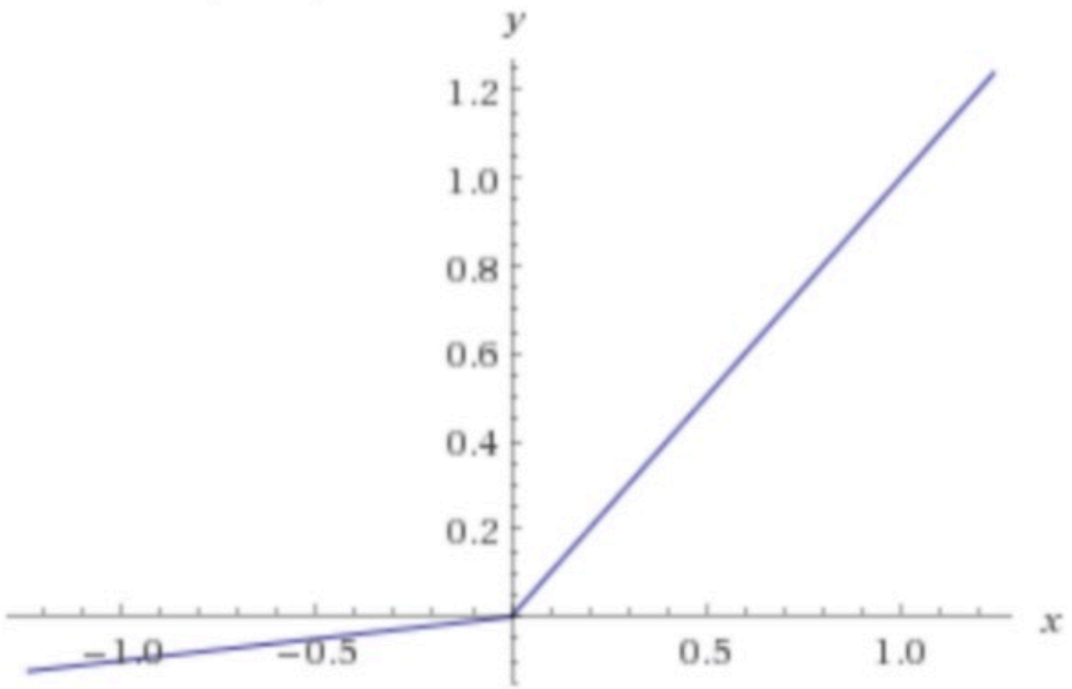
- **Computationally efficient**—allows the network to converge very **quickly**
- **Non-linear**—although it looks like a linear function, ReLU has a derivative function and allows for backpropagation

Cons

- **The Dying ReLU problem**—when inputs approach zero, or are negative, the gradient of the function becomes zero, the network cannot perform backpropagation and cannot learn.

Leaky ReLu Activation Functions

$$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$



Pros

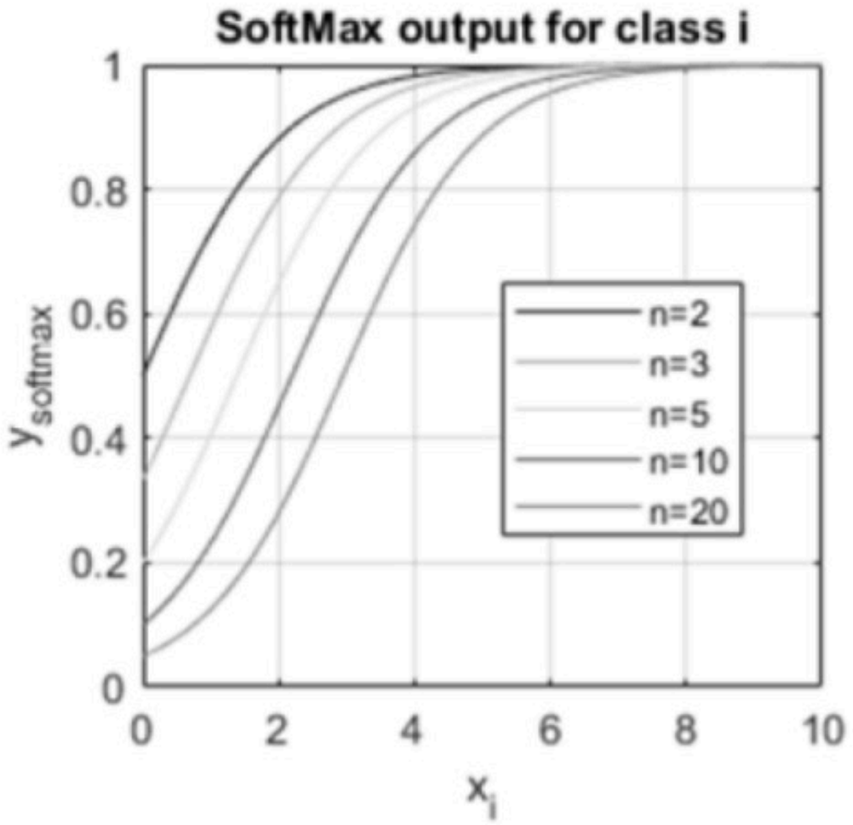
- **Prevents dying ReLU problem**—this variation of ReLU has a small positive slope in the negative area, so it does enable backpropagation, even for negative input values
- Otherwise like ReLU

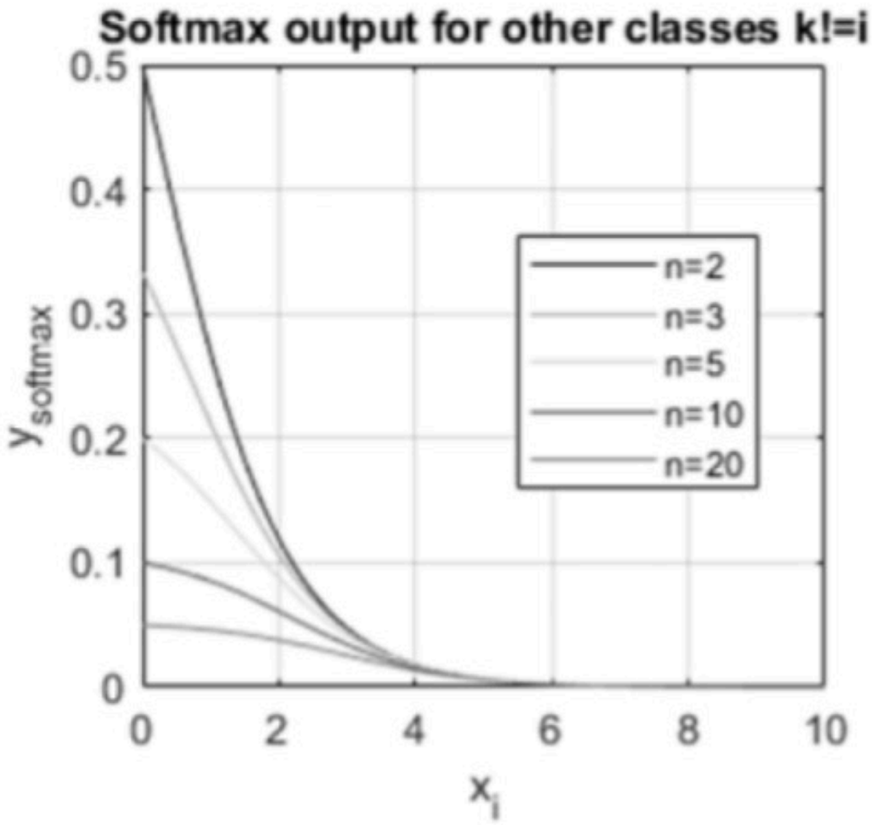
Cons

- **Results not consistent**—leaky ReLU does not provide consistent predictions for negative input values.

Softmax Activation Functions

$$\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$



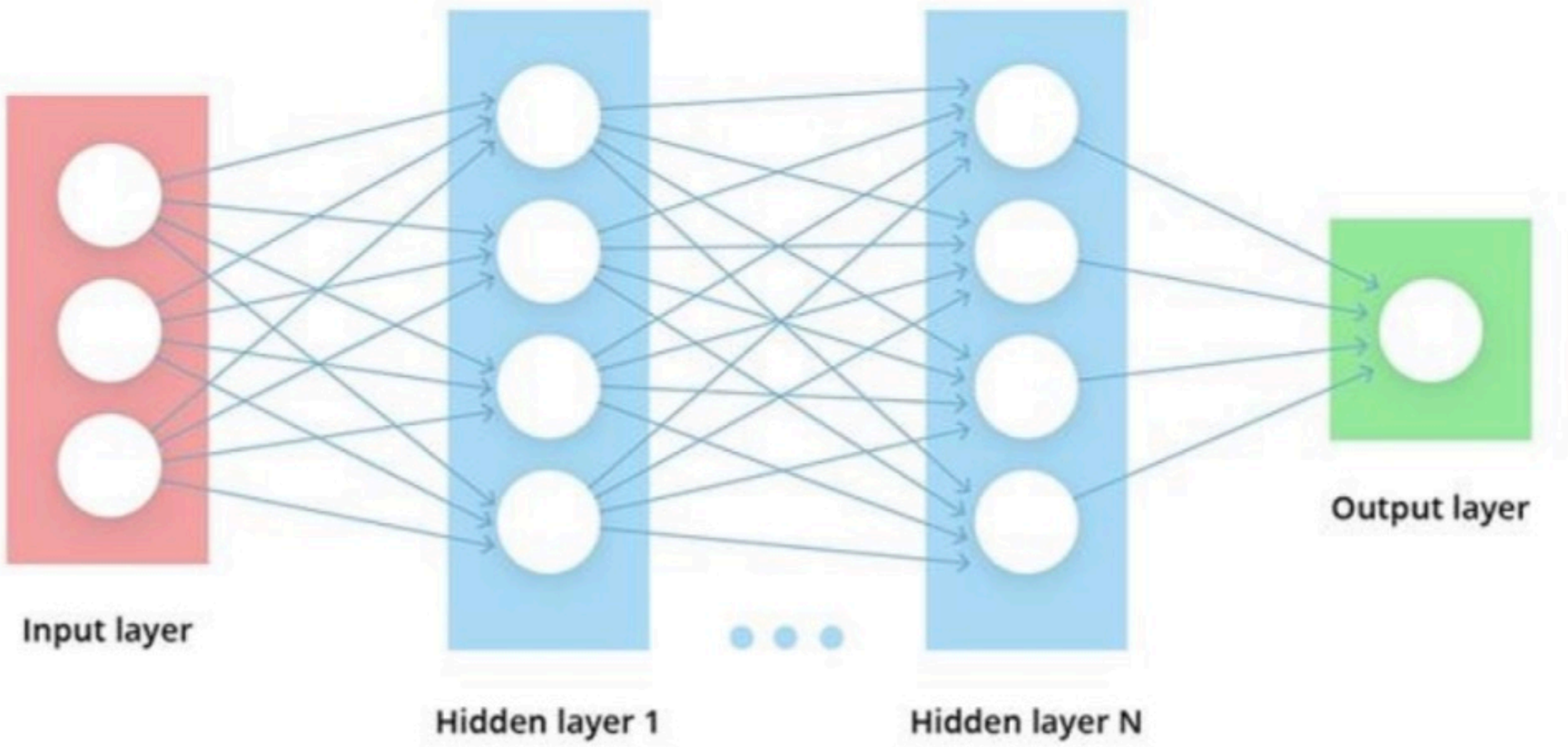


- **Able to handle multiple classes** only one class in other activation functions—normalizes the outputs for each class between 0 and 1

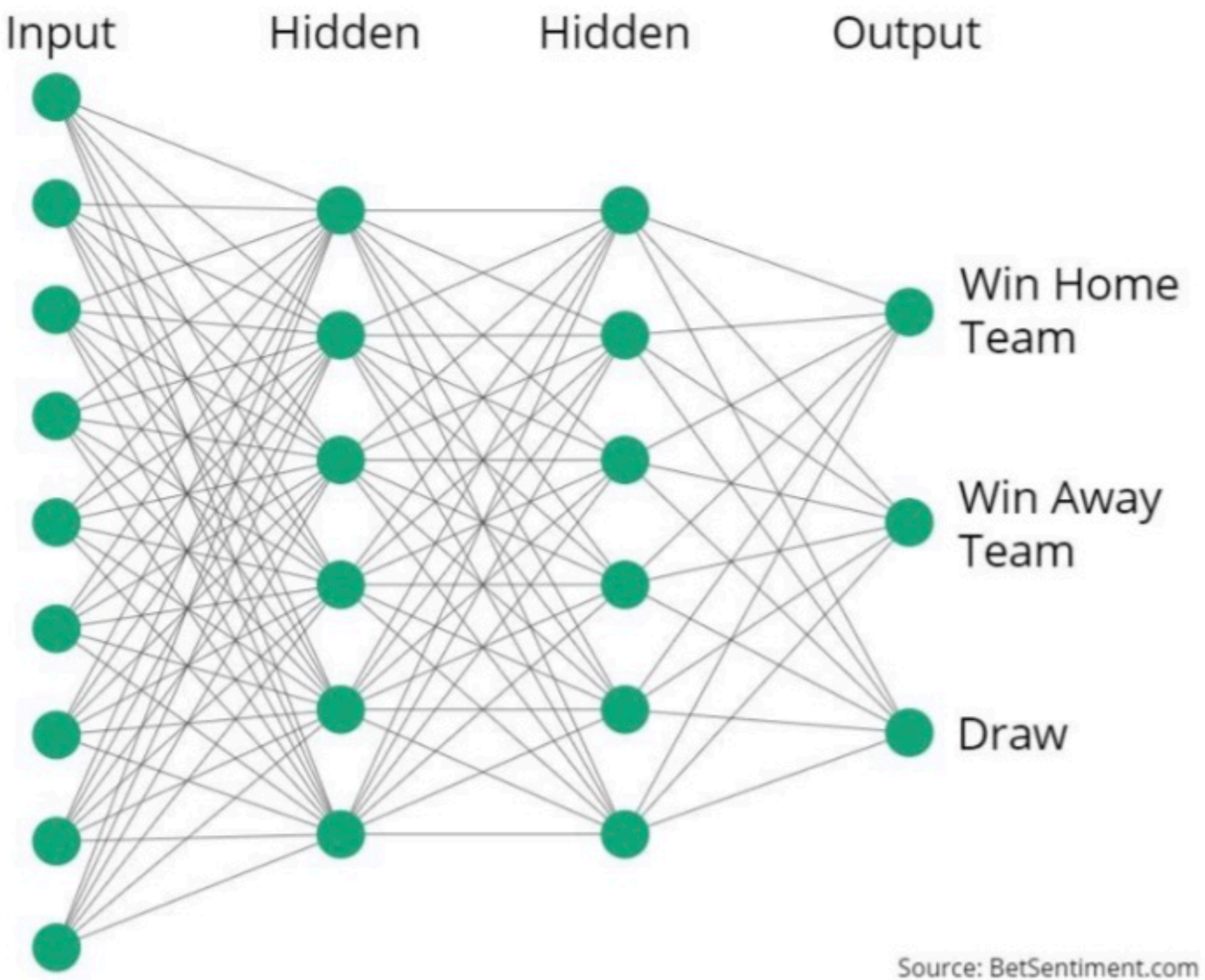
- **Useful for output neurons**—typically Softmax is used only for the output layer

Classification

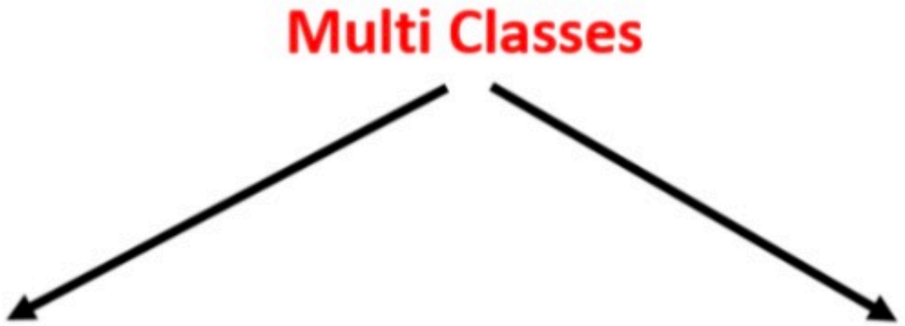
Binary Classification



Multiclass Classification



Multiclass Classification



Mutually Exclusive Classes

Data Point 1	RED
Data Point 2	GREEN
Data Point 3	BLUE
...	...
Data Point N	RED

Softmax Activation Function

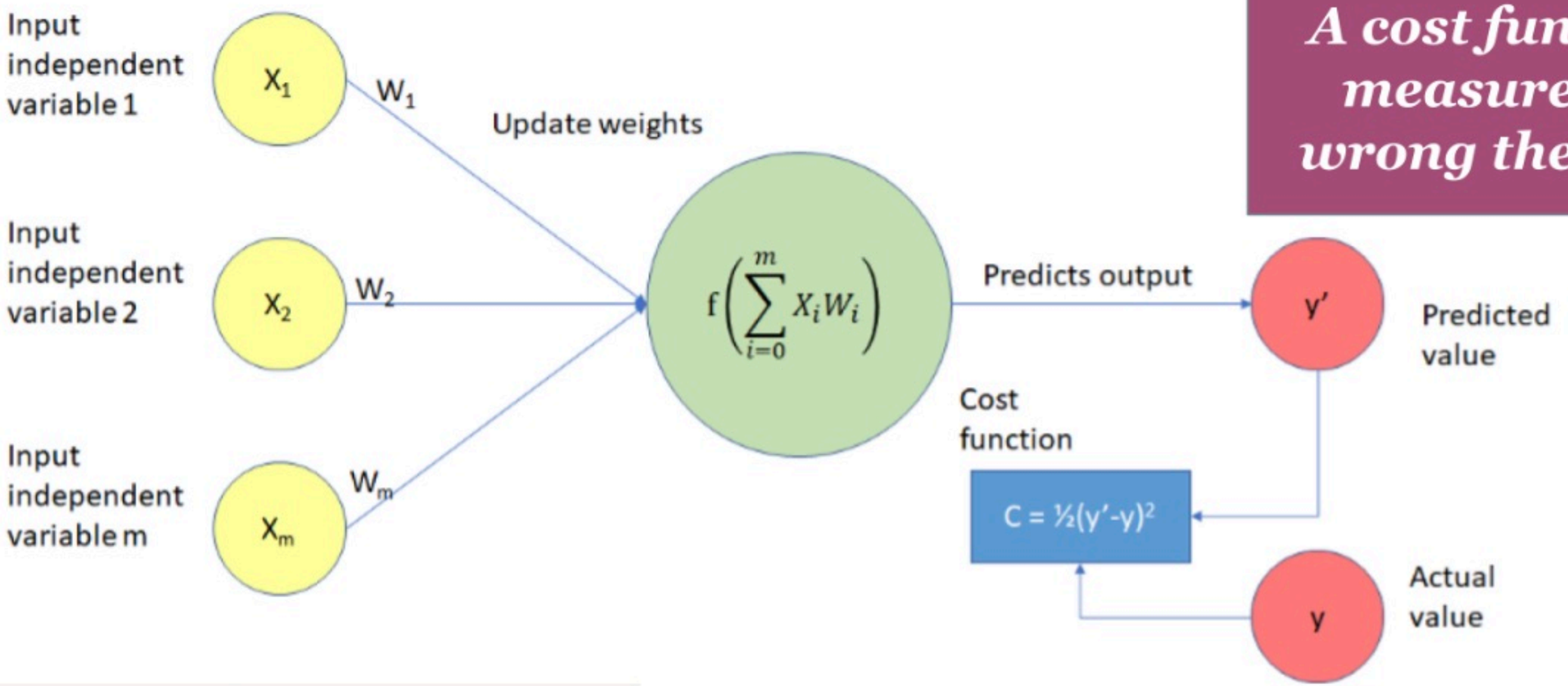
Non-Exclusive Classes

Data Point 1	A,B
Data Point 2	A
Data Point 3	C,B
...	...
Data Point N	B

Sigmoid Activation Function

Cost Function

Cost Function (Loss Function)



A cost function is a measure of how wrong the model is

Cost function $J = \sum_{1}^{m} \frac{1}{2}(y - \hat{y})^2$

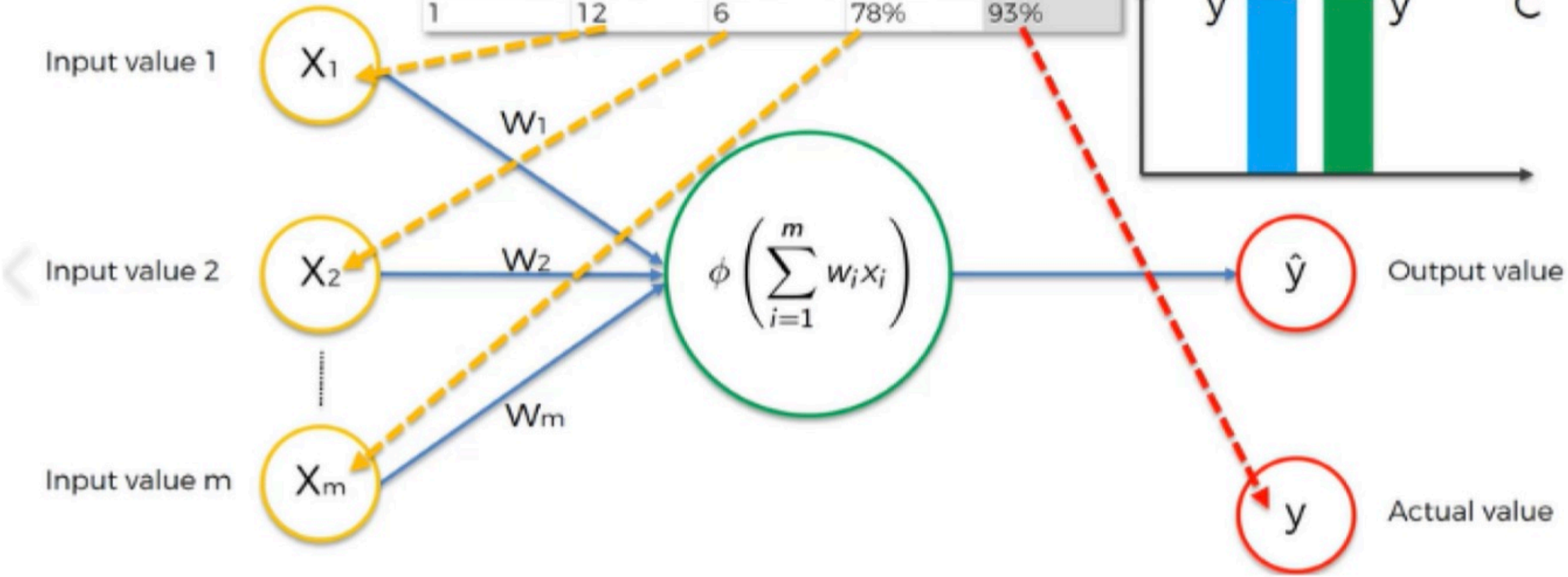
Sum over all samples — true value — predicted value

Cost function (J) = 1/m (Sum of Loss error for 'm' examples)

Cost Function (Loss Function)

Example

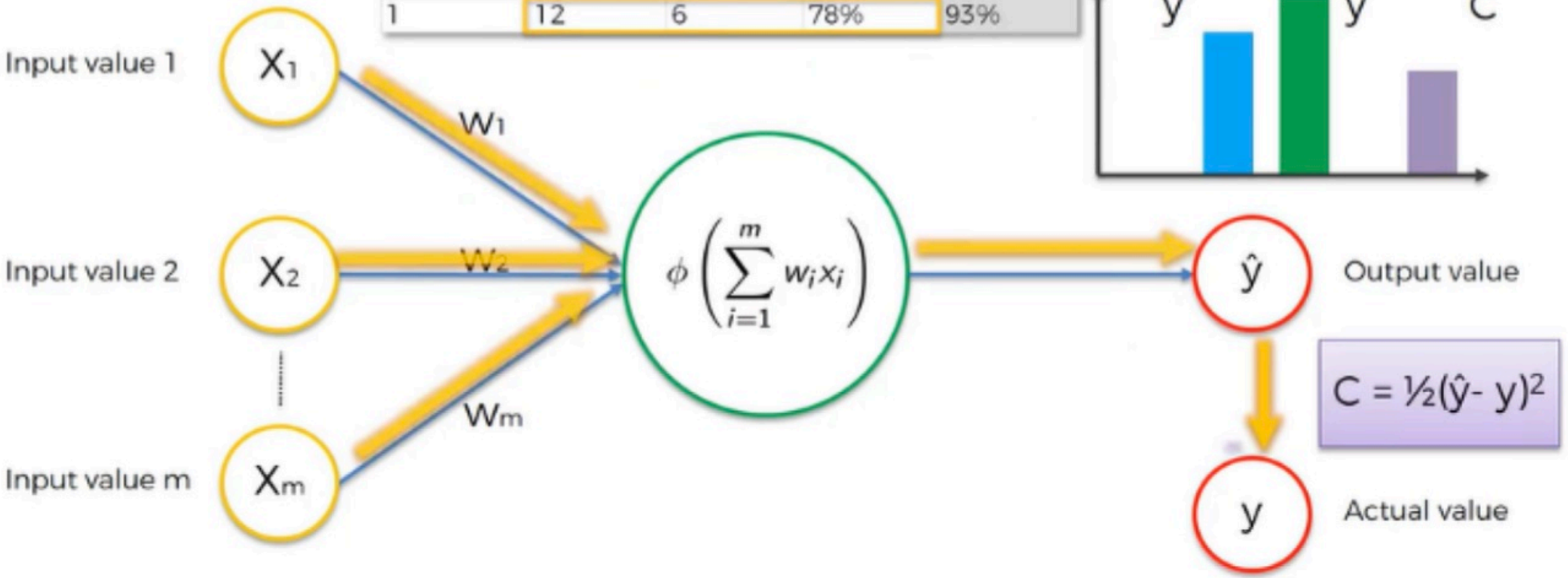
Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%



Cost Function (Loss Function)

Example

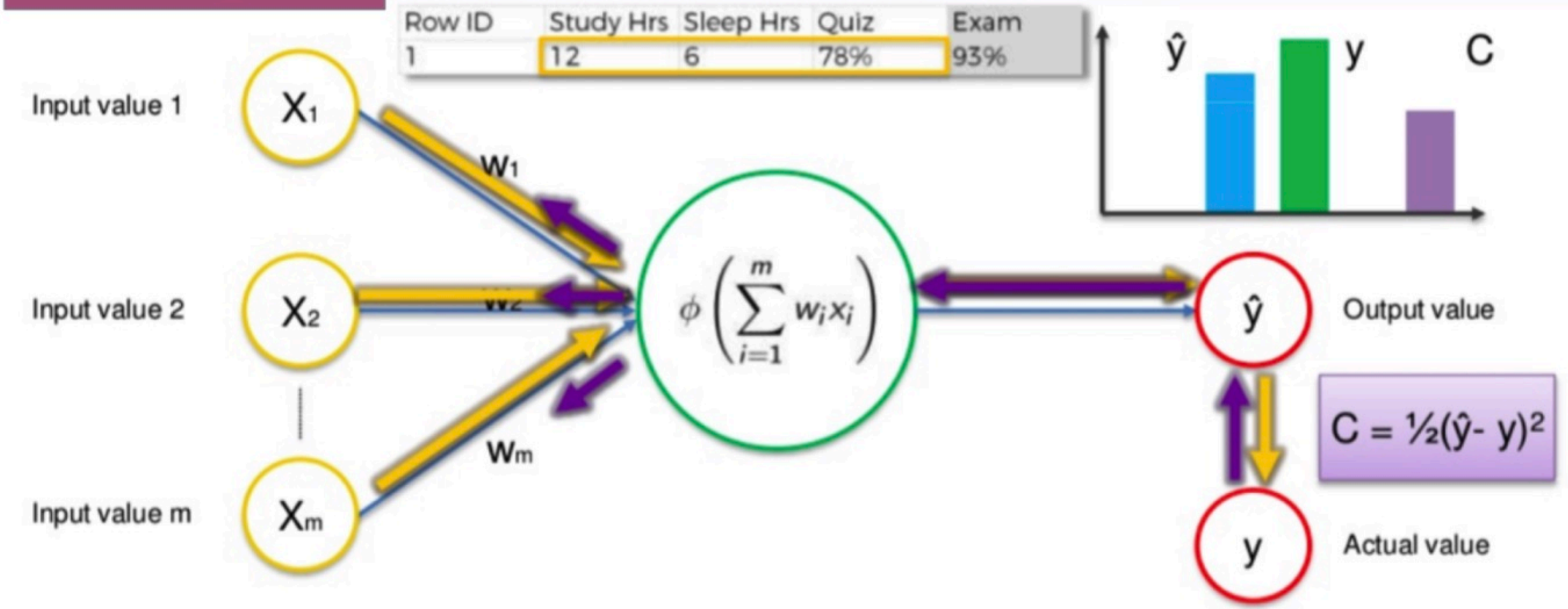
Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%



Cost Function (Loss Function)

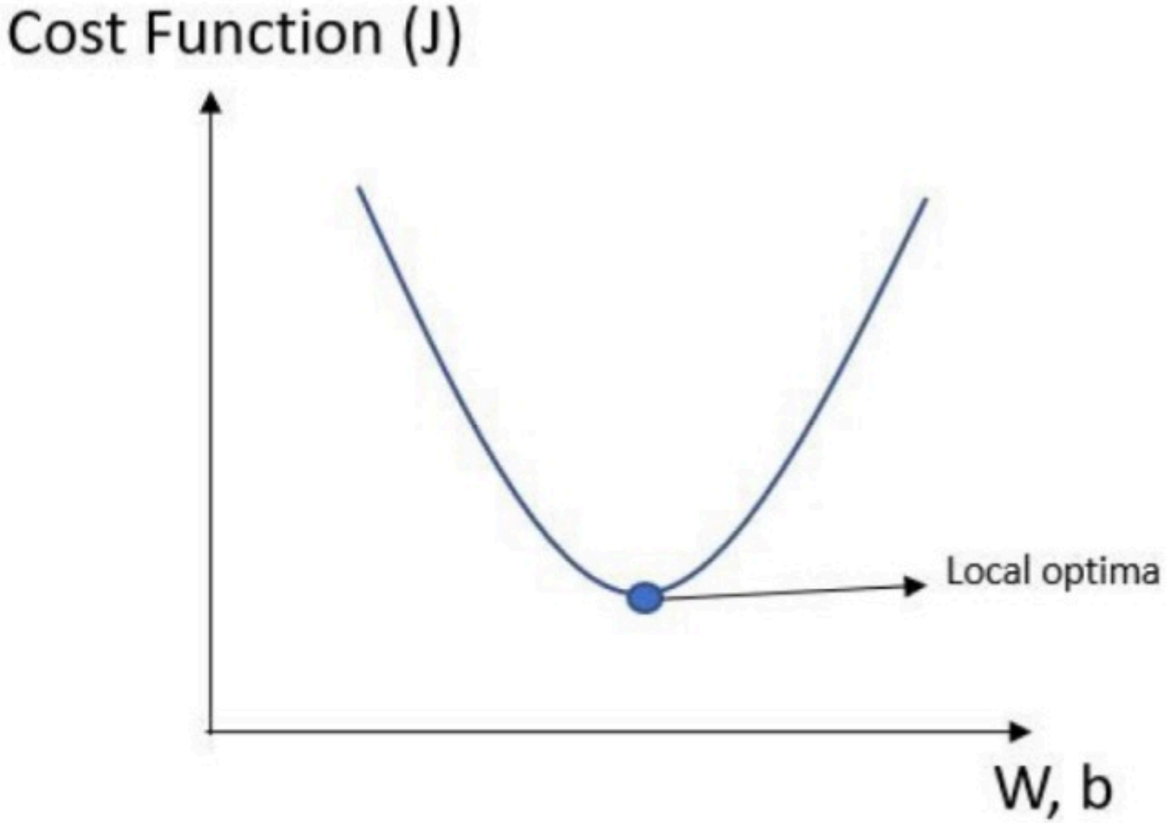
Example

Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%



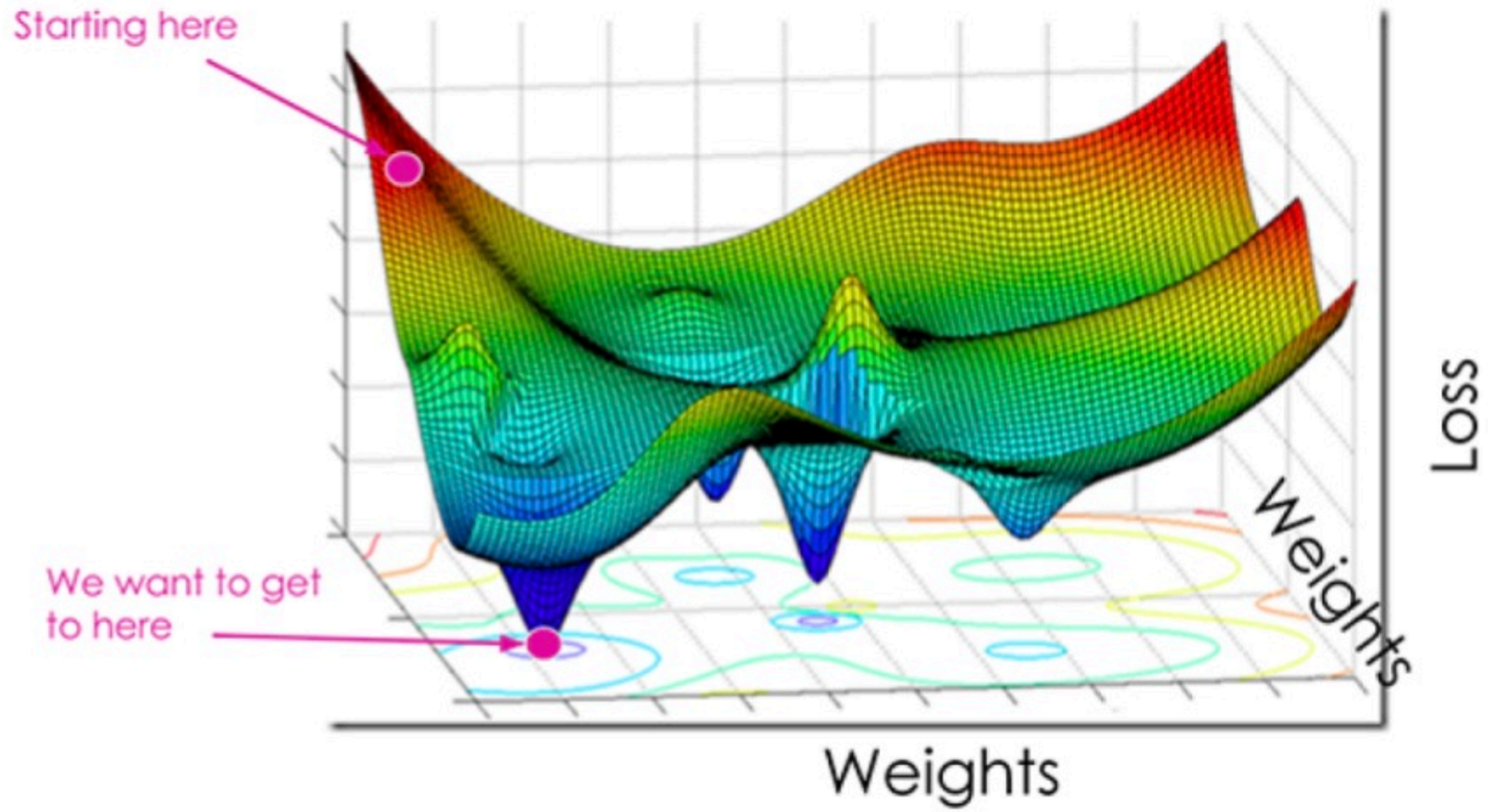
Cost Function (Loss Function)

Cost Function (Loss Function)



The objective of a ML model, is to find parameters, weights or a structure that ***minimises*** *the cost function.*

Cost Function (Loss Function)



The objective of a ML model, is to find parameters, weights or a structure that ***minimises*** *the cost function.*

Types of Cost Function

Regression Loss Functions

```
model.compile(optimizer='adam',loss='mse')
```

1. Mean Squared Error Loss
2. Mean Squared Logarithmic Error Loss
3. Mean Absolute Error Loss

Multi-Class Classification Loss Functions

1. Multi-Class Cross-Entropy Loss
2. Sparse Multiclass Cross-Entropy Loss
3. Kullback Leibler Divergence Loss

```
1 model.compile(loss='categorical_crossentropy',
```

Binary Classification Loss Functions

1. Binary Cross-Entropy
2. Hinge Loss
3. Squared Hinge Loss

```
model.compile(loss='binary_crossentropy', optimizer='adam')
```

Cost Function (Loss Function)

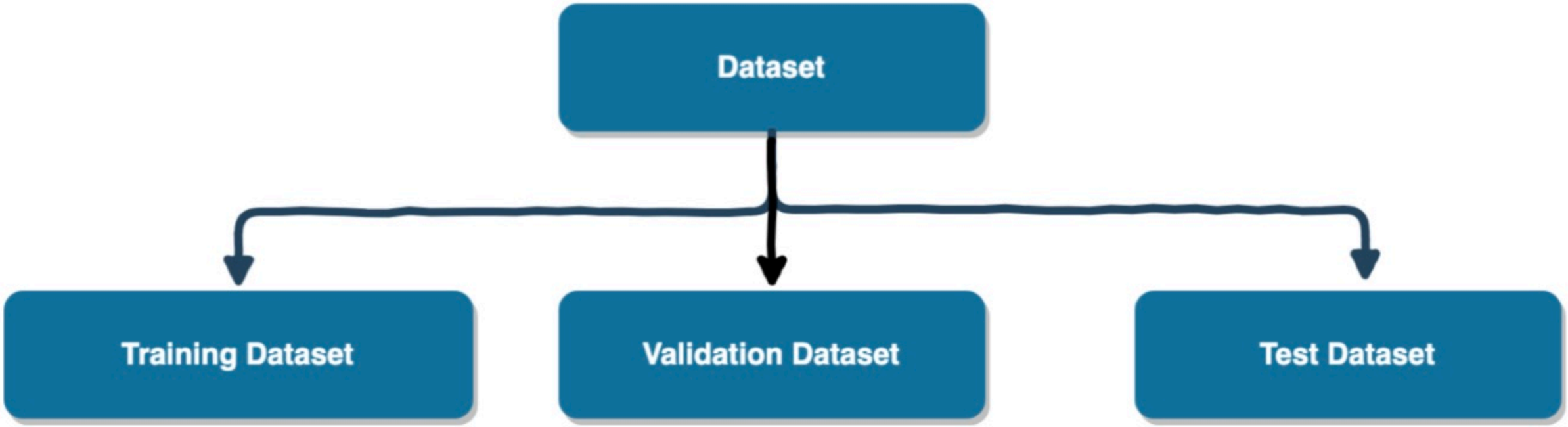
```
model = Sequential()

model.add(Dense(19,activation='relu'))
model.add(Dense(19,activation='relu'))
model.add(Dense(19,activation='relu'))
model.add(Dense(19,activation='relu'))
model.add(Dense(1))

model.compile(optimizer='adam' loss='mse')
```

```
model = Sequential()
model.add(Dense(units=30,activation='relu'))
model.add(Dense(units=15,activation='relu'))
model.add(Dense(units=1,activation='sigmoid'))
model.compile loss='binary_crossentropy' , optimizer='adam')
```

Cost Function (Loss Function)




```
model.fit(x = X_train, y = y_train,
          validation_split = 0.1,
          batch_size = 32,
          epochs = 1000, verbose=1)
```

Cost Function (Loss Function)

```
Epoch 1/600
426/426 [==============================] - 0s 1ms/sample - loss: 0.6807 - val_loss: 0.6623
Epoch 2/600
426/426 [==============================] - 0s 108us/sample - loss: 0.6498 - val_loss: 0.6330
Epoch 3/600
426/426 [==============================] - 0s 80us/sample - loss: 0.6188 - val_loss: 0.6006
```

■

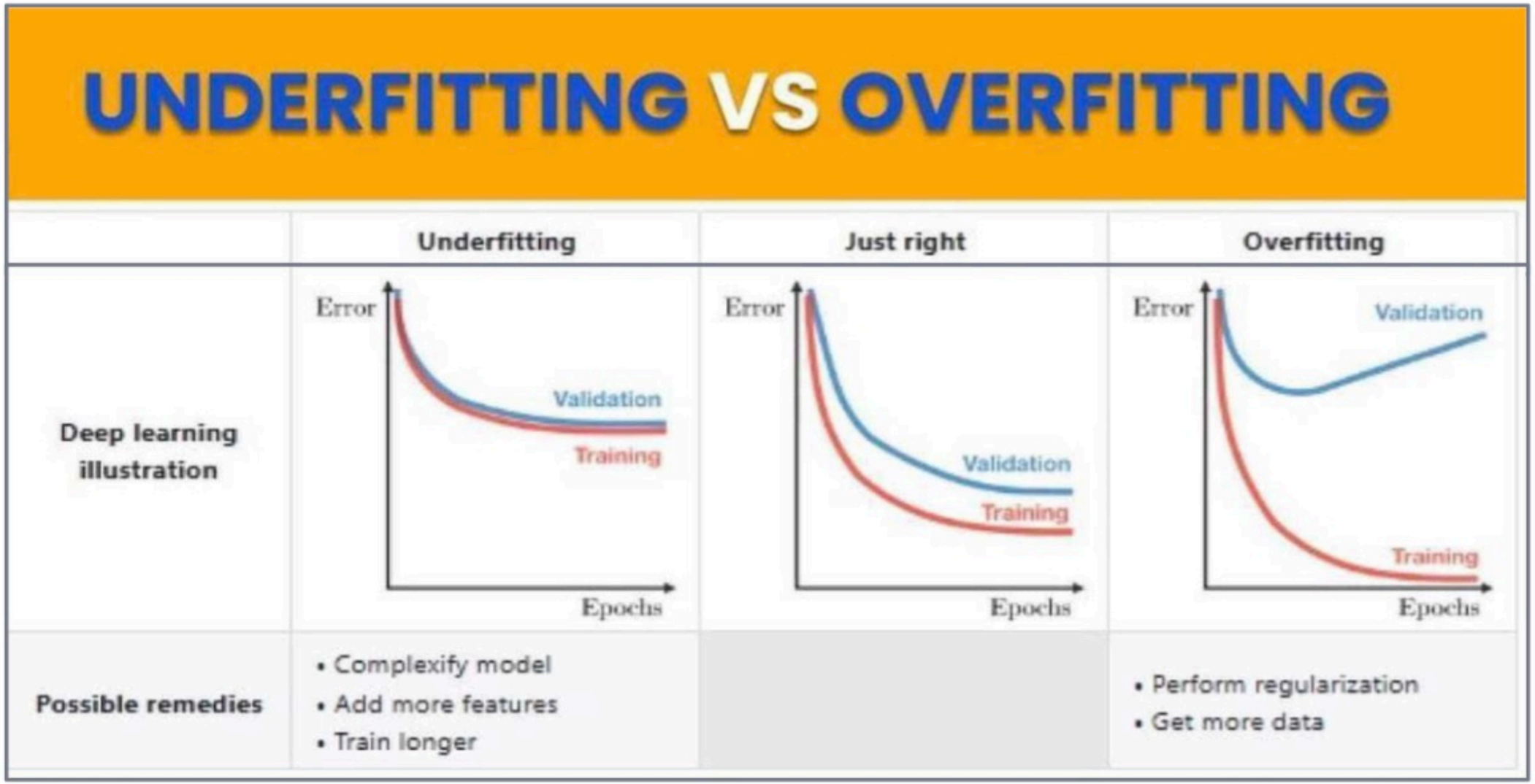
■

■

```
Epoch 99/600
426/426 [==============================] - 0s 68us/sample - loss: 0.0485 - val_loss: 0.1003
Epoch 100/600
426/426 [==============================] - 0s 73us/sample - loss: 0.0483 - val_loss: 0.1060
Epoch 101/600
426/426 [==============================] - 0s 68us/sample - loss: 0.0500 - val_loss: 0.1036
Epoch 00101: early stopping
```

decreases as model learns

Cost Function (Loss Function)



Cost Function (Loss Function)

Problem Type	Output Type	Final Activation Function	Loss Function
Regression	Numerical value	Linear	Mean Squared Error (MSE)
Classification	Binary outcome	Sigmoid	Binary Cross Entropy
Classification	Single label, multiple classes	Softmax	Cross Entropy
Classification	Multiple labels, multiple classes	Sigmoid	Binary Cross Entropy