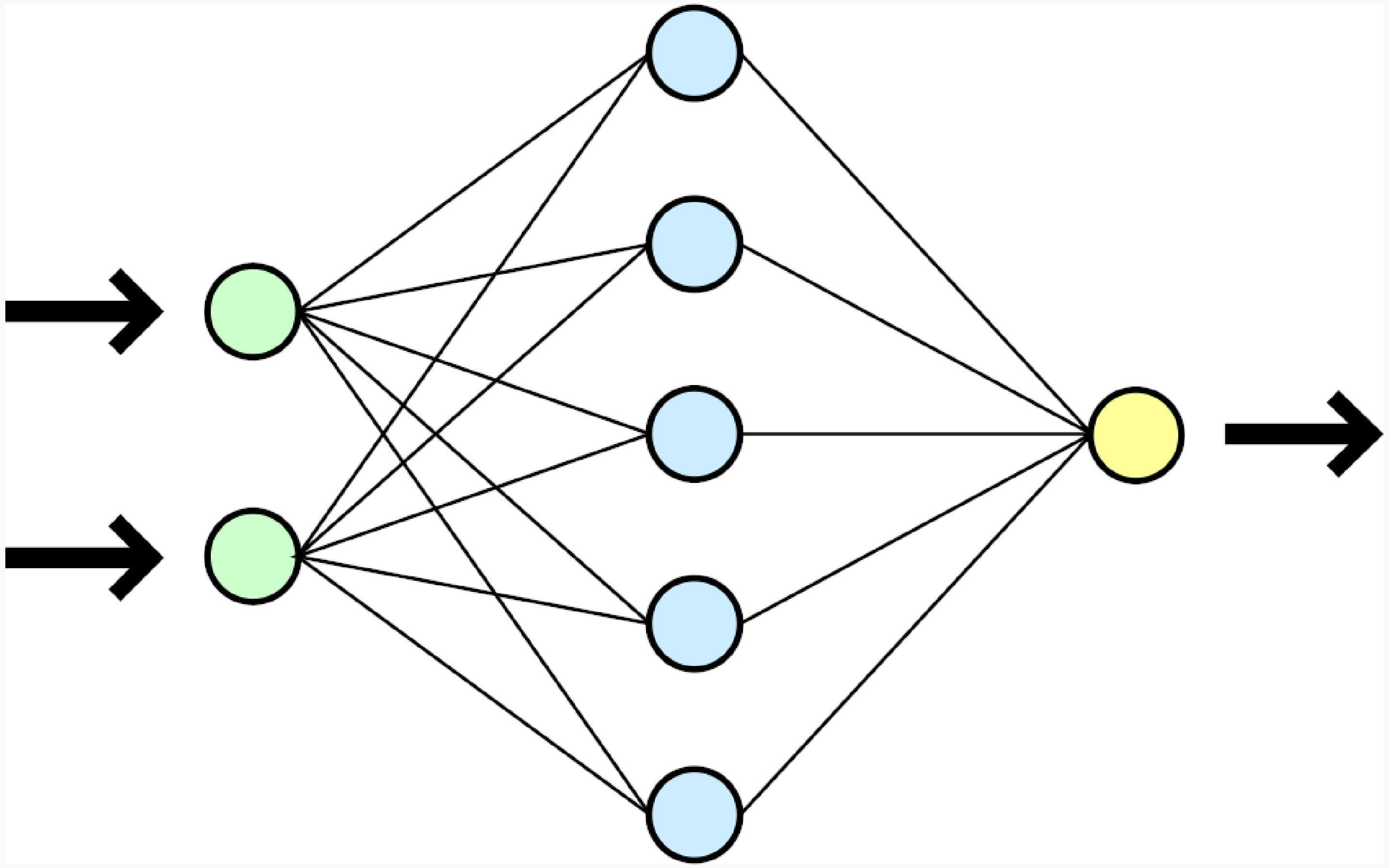
SIAD M2 DS – 2023/2024

Machine Learning – Neural Network

Nadarajen Veerapen

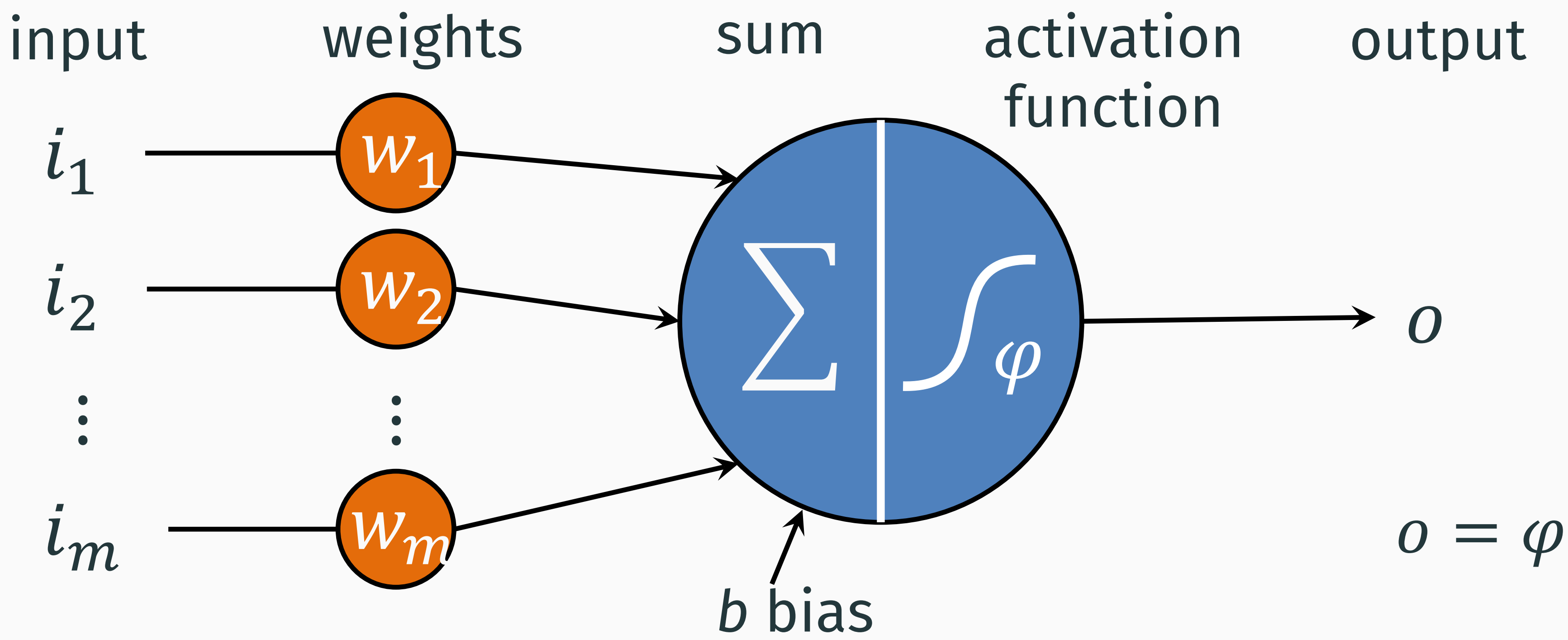
Faculté des sciences économiques, sociales et des territoires – Université de Lille

Artificial Neural Networks



Artificial Neural Networks (ANNs) are systems inspired by the way biological neurons work. They are built around the concept of artificial neurons (*neurone formel)*. They can be used for both classification and regression.

Artifical Neuron



$$o = \varphi\left(b + \sum_{j=1}^{m} w_j i_j\right)$$

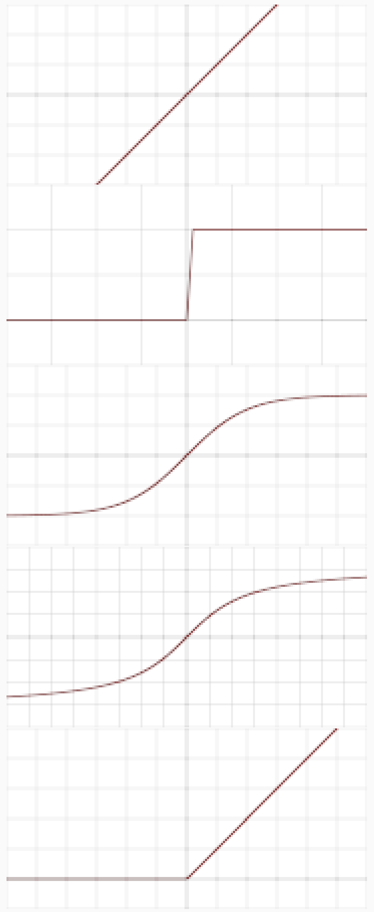
An artificial neuron is a simplified mathematical representation of a biological neuron. In the general case, an artifical neuron has m inputs which are associated to an output via an mathematical operation.

Each input i_j is associated to a weight w_j (synaptic coefficients). The first operation is a weighted sum of the inputs. The result is transformed using a non-linear *activation fonction* φ (phi). The bias (b) is an optional constant term.

Activation Functions

The activation function is used to introduce non-linearity in the model model.

If the function was linear for all neurons in a network, then the overall computation would also be linear (and thus only one neuron would be sufficient to model the same thing).



Examples:

Identity function (linear), $f(x) = x$

Step function, $f(x) = \begin{cases} 0 \text{ si } x < 0 \\ 1 \text{ si } x \geq 0 \end{cases}$

Hyperbolic Tangent function (TanH), $f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$

Arc Tangent function (ArcTan), $f(x) = \tan^{-1} x$

Rectified Linear Unit function (ReLU), $f(x) = \begin{cases} 0 \text{ si } x < 0 \\ x \text{ si } x \geq 0 \end{cases}$

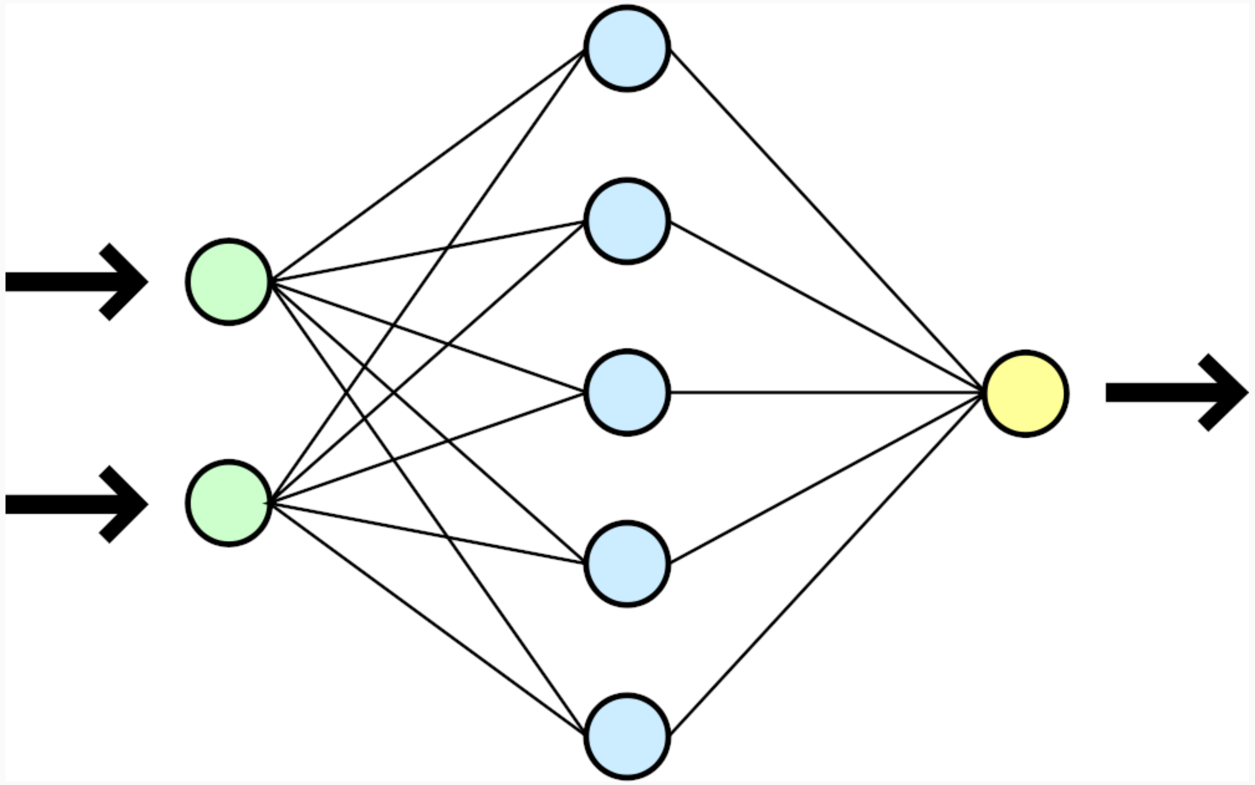
ReLU now commonly used, especially in deep neural networks.

Multi-Layer Perceptron

The multi-layer perceptron (MLP) is a type of artificial neural network where the neurons (or nodes) are organized in layers:

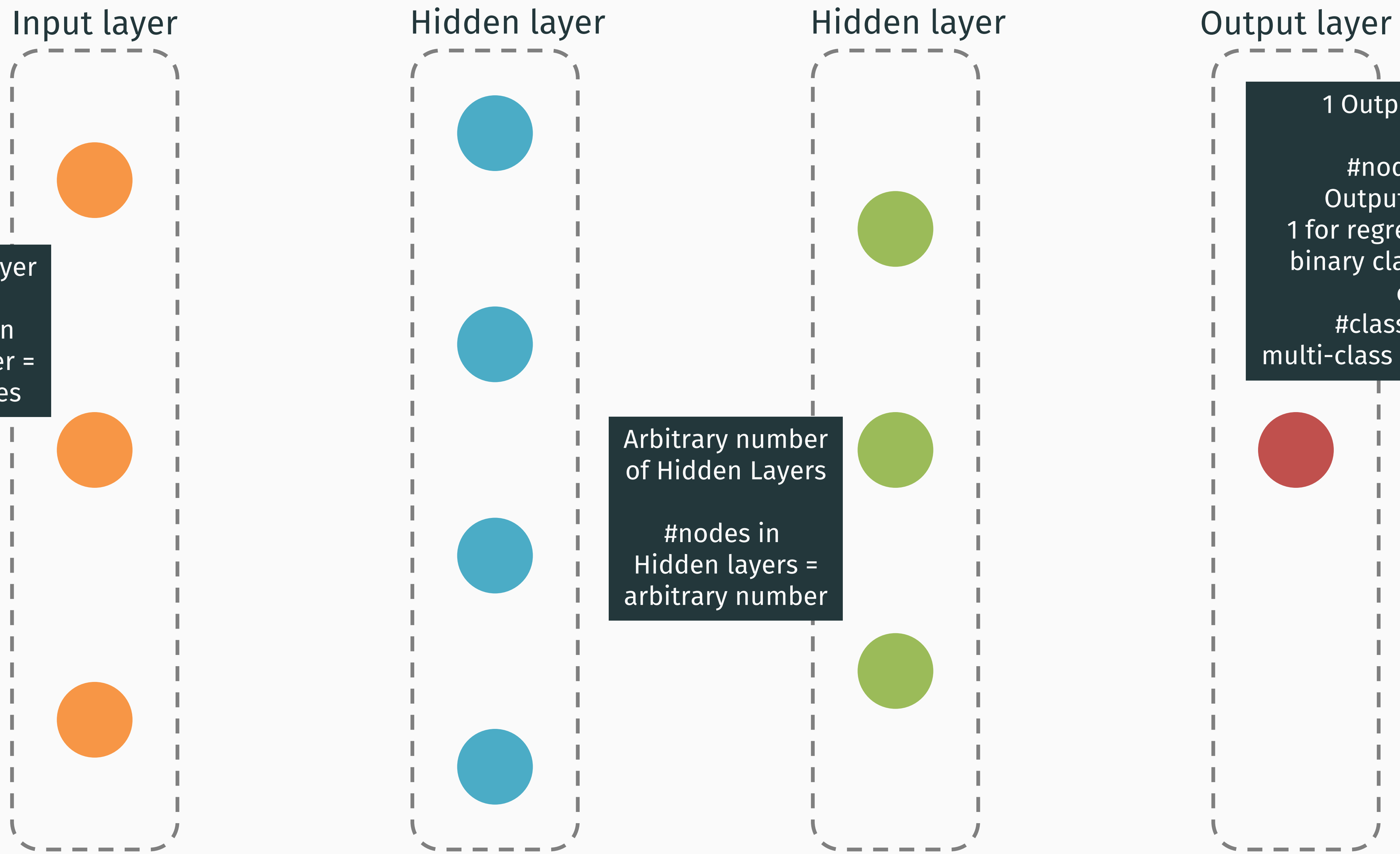
- The input layer
- The hidden layers
- The output layer

Information travels from the input layer to the output layer.
The neurons in a layer are not connected between themselves but are connected to the neurons of the adjacent layers.



Tensorflow Playground

Multi-Layer Perceptron Layout



1 Input Layer

#nodes in
Input layer =
#variables

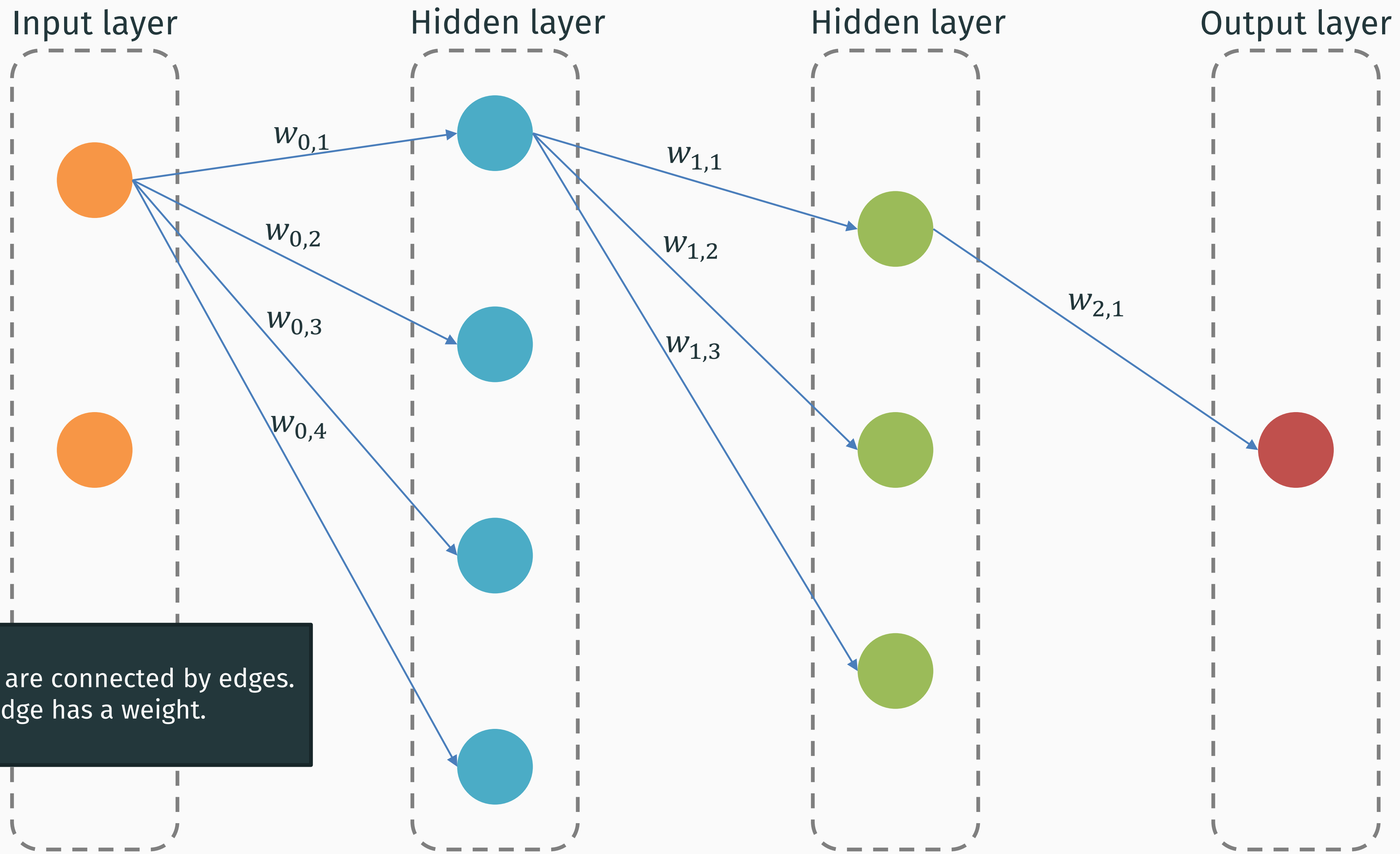
Arbitrary number
of Hidden Layers

#nodes in
Hidden layers =
arbitrary number

1 Output Layer

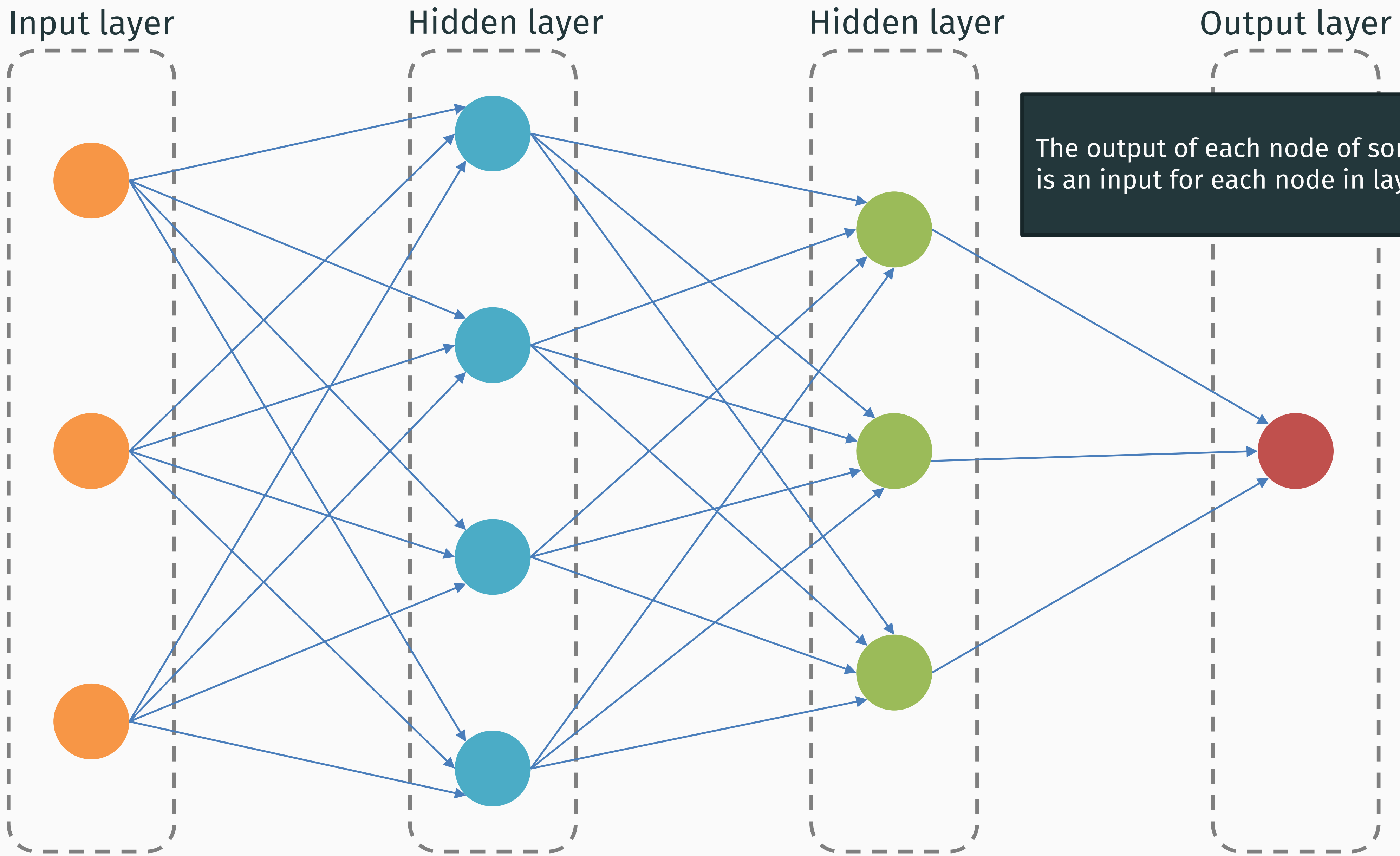
#nodes in
Output layer =
1 for regression and
binary classification
or
#classes for
multi-class classification

Multi-Layer Perceptron Layout



Nodes are connected by edges.
Each edge has a weight.

Multi-Layer Perceptron Layout



The output of each node of some layer *i* is an input for each node in layer *i+1*.

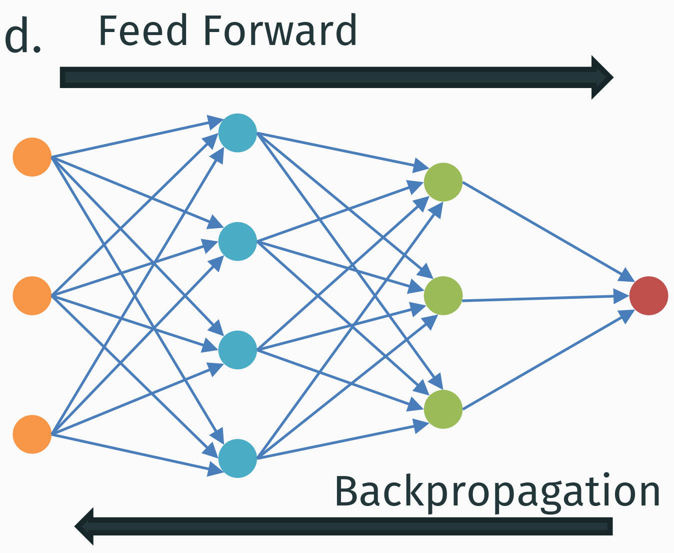
Training an ANN

Initialization
At the beginning, weights are usually generated randomly to relatively small values, biases as well.

Forward Pass/Feed Forward
Layer by layer, weights and inputs are used to compute the output of each node.
The result is passed forward to the next layer until the output layer is reached.

Calculating the Error
The difference between the true value and the output value of the network is computed.

Reducing the Error
The goal is to reduce the error. This is performed by changing the weights throughout the network. This is done using *backpropagation*.



Backpropagation
Update the weights, starting from the last layer and going backwards. Each weight is going to be changed according to a local error that is minimized using gradient descent:

$w'_{ij} = w_{ij} + \Delta w_{ij}$ where $\Delta w_{ij} = -\eta \frac{\partial Error}{\partial w_{ij}}$ and η is called the *learning rate*

Repeat Feed Fordward and Backpropagation until convergence or maximum iterations/epochs reached.

Considerations for MLPs and ANNs

- Able to learn non-linear, non-polynomial relationships.
- Able to create very complex models (GPT 3 has 96 layers and 175 billion parameters).
- Training the model can take a lot of time depending on the size of the network.
- Better to have data that are fairly homogeneous, ideally the data should have a mean close or equal to 0 and a variance of 1.
- Optimizing the coefficients (with gradient descent) does not guarantee that the best set of coefficients will be found. Using a different seed for the pseudo-random number generator may give a different result.

Considerations for MLPs and ANNs

Many parameters to optimize, including:

- The **number of hidden layers** (one strategy: test an increasing number of hidden layers until there is no improvement)
- The **number of nodes per layer** (one strategy: pick something between the size of the input layer and that of the output layer; have a decreasing number of nodes in each successive hidden layer)
- The **activation function**(s)
- The **regularization** parameter *alpha* α that penalizes coefficients
- The **learning rate** *eta* η that defines the strength of learning
- The **solver** used to compute the coefficients (the solver itself will also have parameters)
- The **number of iterations** or **epochs** used to compute the coefficients.