CS 453X: Class 18

Jacob Whitehill

More on *k*-means

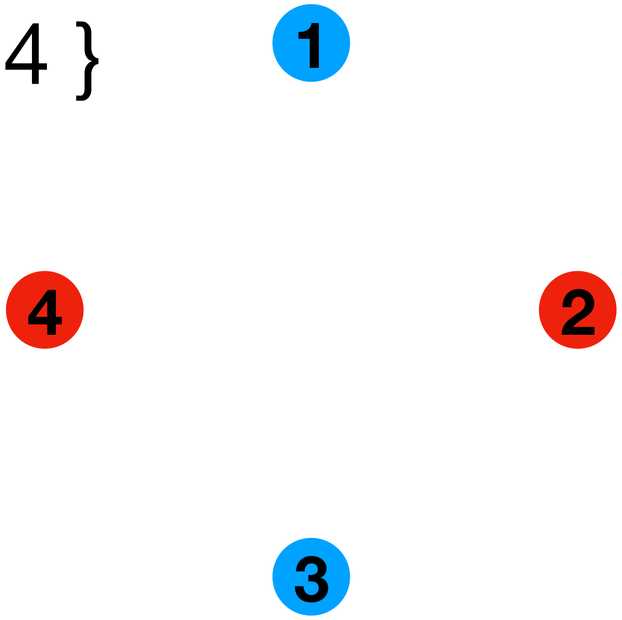
Exercise: Empty clusters (1)

- Assume that a set of *distinct* data points { $\mathbf{x}^{(i)}$ } are initially assigned so that none of the *k* clusters is empty.

- How can it happen that, during execution of the *k*-means algorithm, a cluster becomes empty?

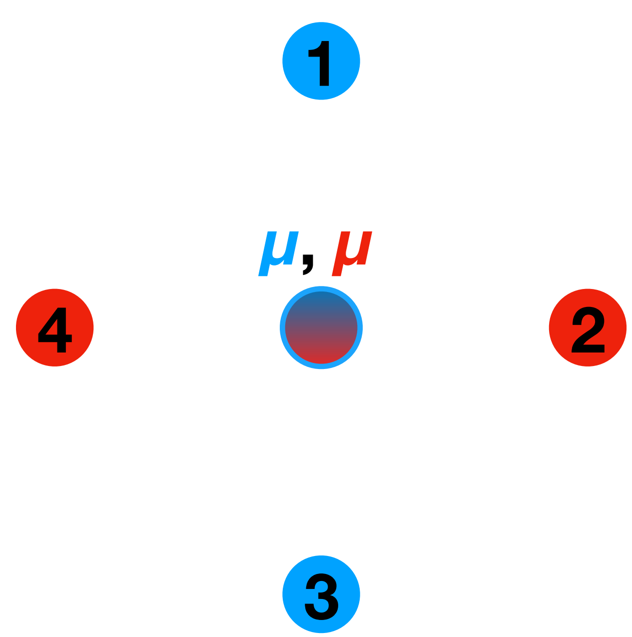
Exercise: Empty clusters (1)

- Example:
  - Blue cluster: { 1, 3 }
  - Red cluster: { 2, 4 }



Exercise: Empty clusters (1)

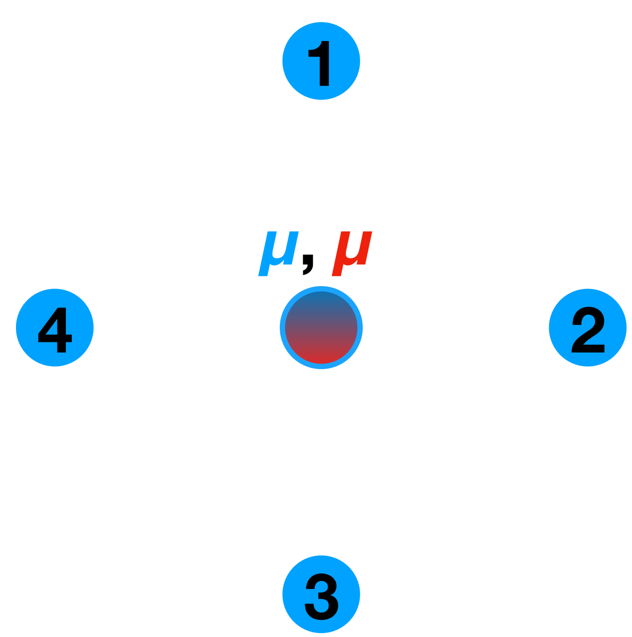
- Example:
  - Blue mean = Red mean



Exercise: Empty clusters (1)

- Example:
  - Blue cluster: { 1, 2, 3, 4 }
  - Red cluster: { }

This was arbitrary. It could also be the reverse.



Exercise: Empty clusters (2)

- Suppose k-means on a set of n *distinct* data points $\{ \mathbf{x}^{(i)} \}$, where $k \leq n$, returns at least one cluster that is empty.

- Can that clustering be globally optimal?

$$\sum_{j=1}^{k} \sum_{i:a(\mathbf{x}^{(i)})=j} \left(\mathbf{x}^{(i)} - \mathbf{c}^{(j)}\right)^2$$

Exercise: Empty clusters (2)

- Suppose *k*-means on a set of *n* *distinct* data points { $\mathbf{x}^{(i)}$ }, where $k \leq n$, returns at least one cluster that is empty.

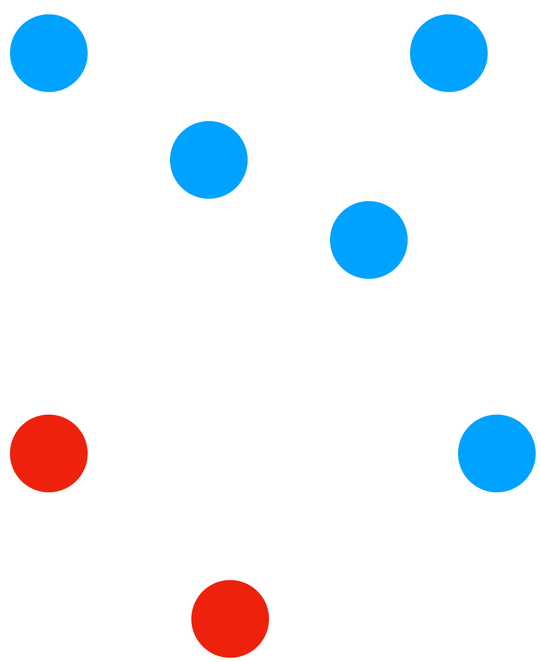
- Can that clustering be globally optimal?

No.

$$\sum_{j=1}^{k} \sum_{i:a(\mathbf{x}^{(i)})=j} \left(\mathbf{x}^{(i)} - \mathbf{c}^{(j)}\right)^2$$

Exercise: Empty clusters (2)

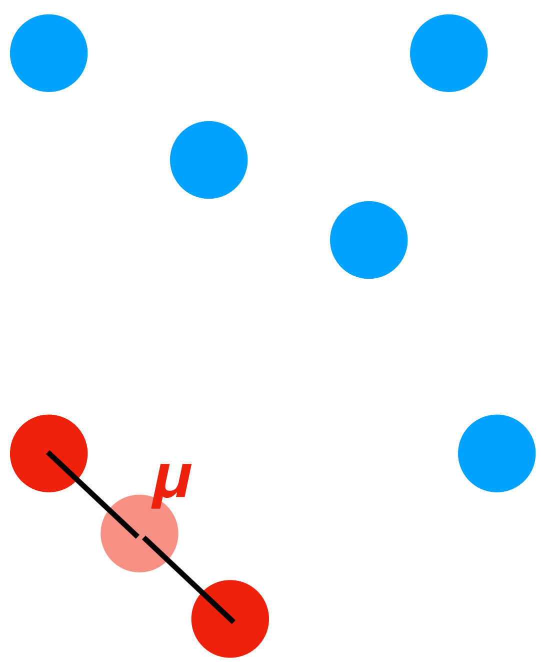
- Reason:
    - Since there are $n \geq k$ data points, and since there were fewer than k clusters returned, then we know there is at least one cluster c with more than 1 data point.



Example: n=7, k=3, but one cluster empty.

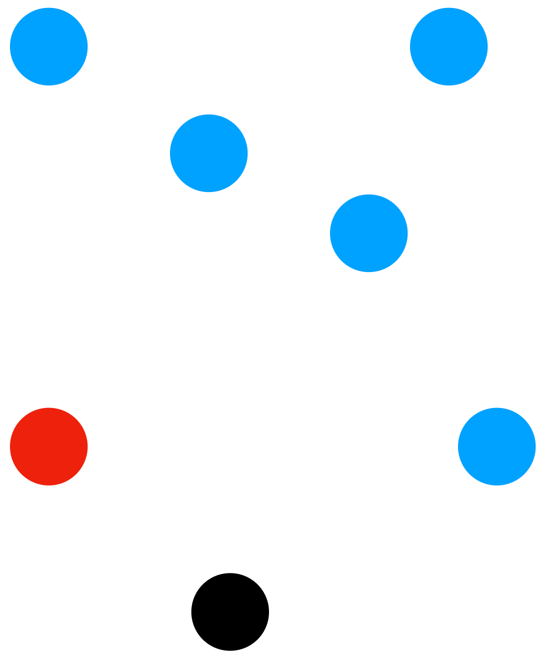
Exercise: Empty clusters (2)

- Reason:
  - Consider the data assigned to cluster *c*. Since they are distinct, then the distance from each point to the cluster mean must be positive.



Exercise: Empty clusters (2)

- Reason:
  - We can reduce the overall cost by creating another cluster (so we have k total) by arbitrarily assigning one of the data in c to the new cluster.



Neural networks

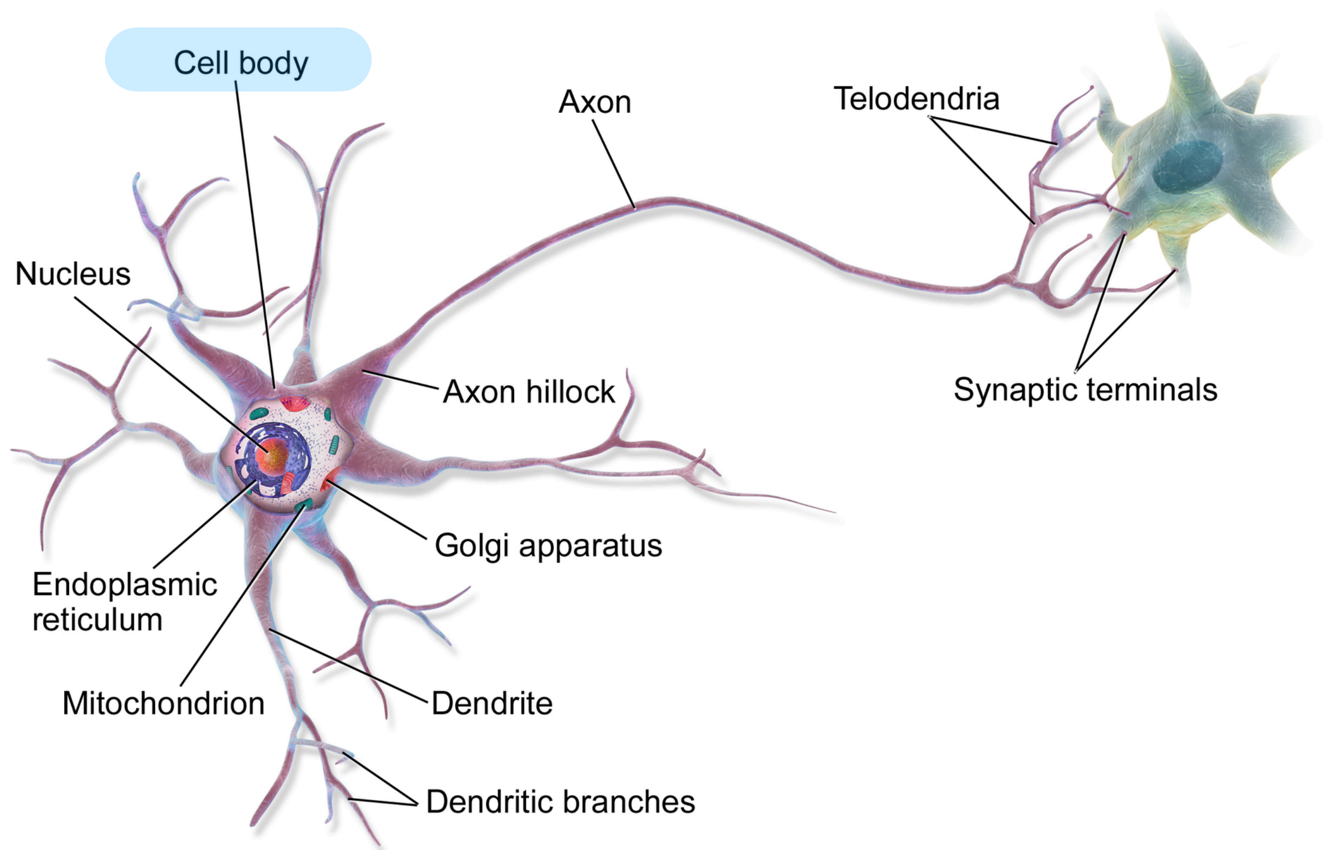
Neural networks

- Since ~2008, the ML model that has made the largest impact on the field is the (resurgence of) the neural network.
- Artificial **neural networks** (**NN**) have existed since the 1940s.
- Over the years, their prominence has waxed and waned, as scientists have alternately made new breakthroughs or run into new roadblocks.
- The field of **deep learning** — based on much *computationally deep* neural networks than was previously possible— has emerged since around 2008.

Biological neural networks

- In biological systems, neural networks consist of a network of connected **neurons**, i.e., brain cells.

- Neurons consist of several components:
  - Soma (cell body)

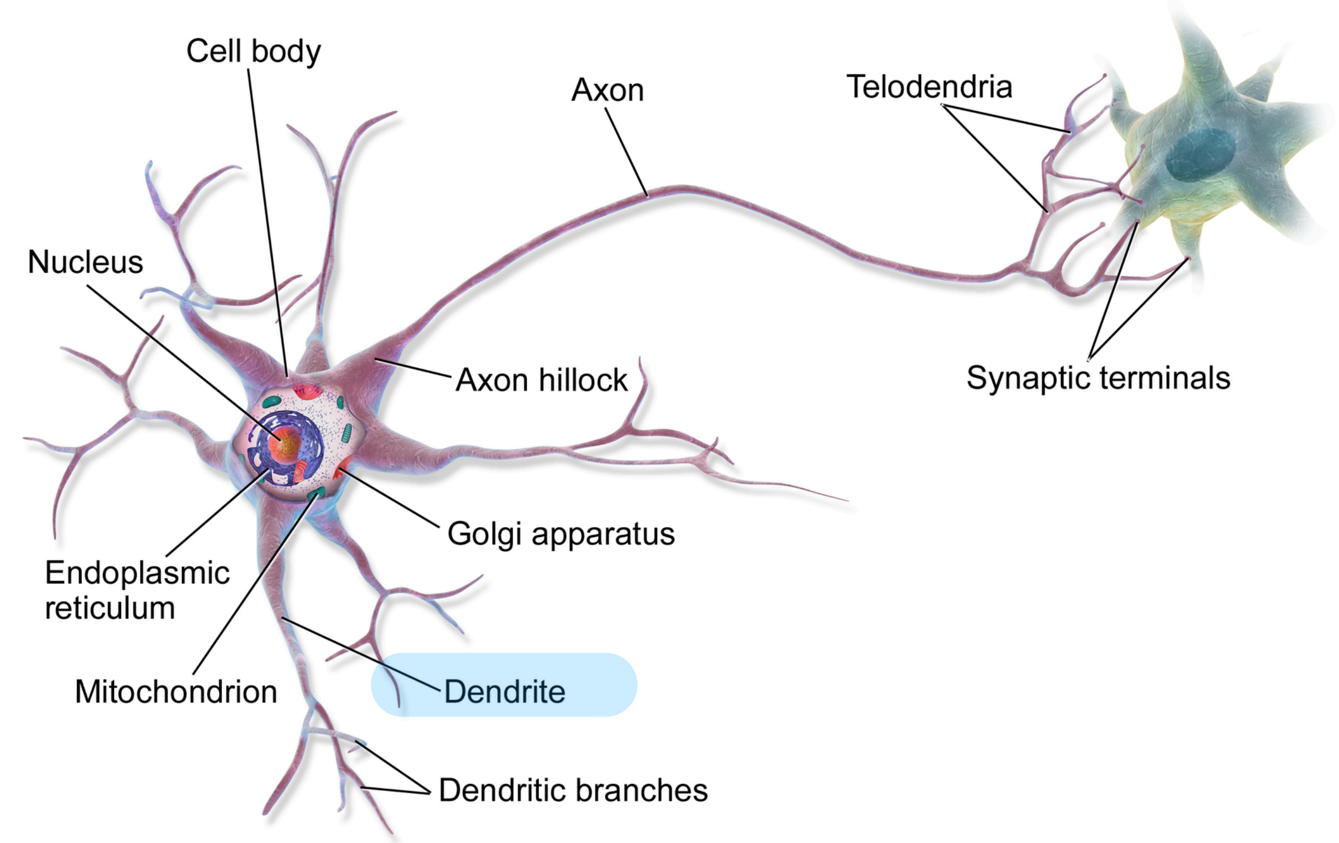


https://en.wikipedia.org/wiki/Neuron#/media/File:Blausen_0657_MultipolarNeuron.png

Biological neural networks

- In biological systems, neural networks consist of a network of connected **neurons**, i.e., brain cells.

- Neurons consist of several components:
  - Soma (cell body)
  - Dendrites (receptors from other neurons)

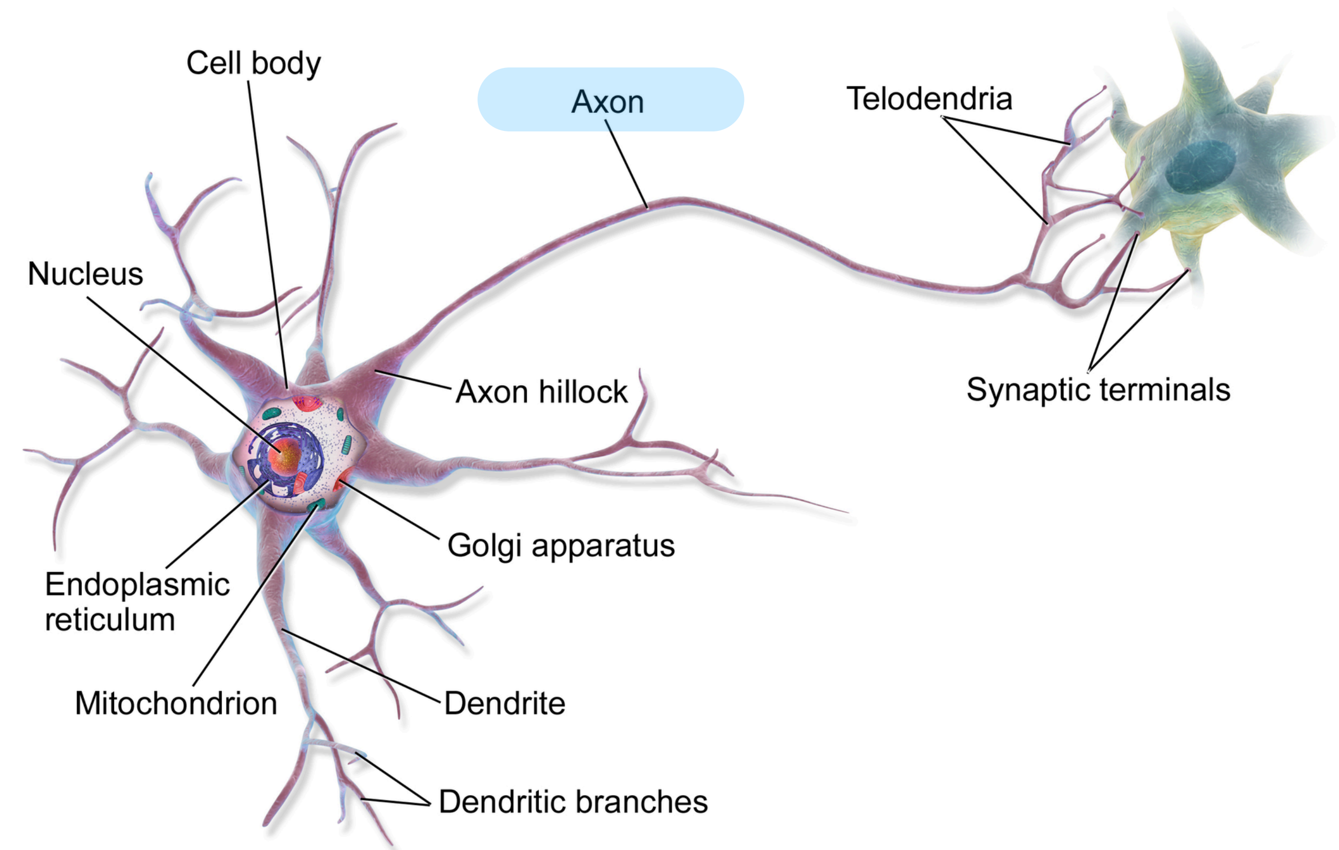


https://en.wikipedia.org/wiki/Neuron#/media/File:Blausen_0657_MultipolarNeuron.png

Biological neural networks

- In biological systems, neural networks consist of a network of connected **neurons**, i.e., brain cells.

- Neurons consist of several components:
    - Soma (cell body)
    - Dendrites (receptors from other neurons)
    - Axons (transmitters to other neurons)

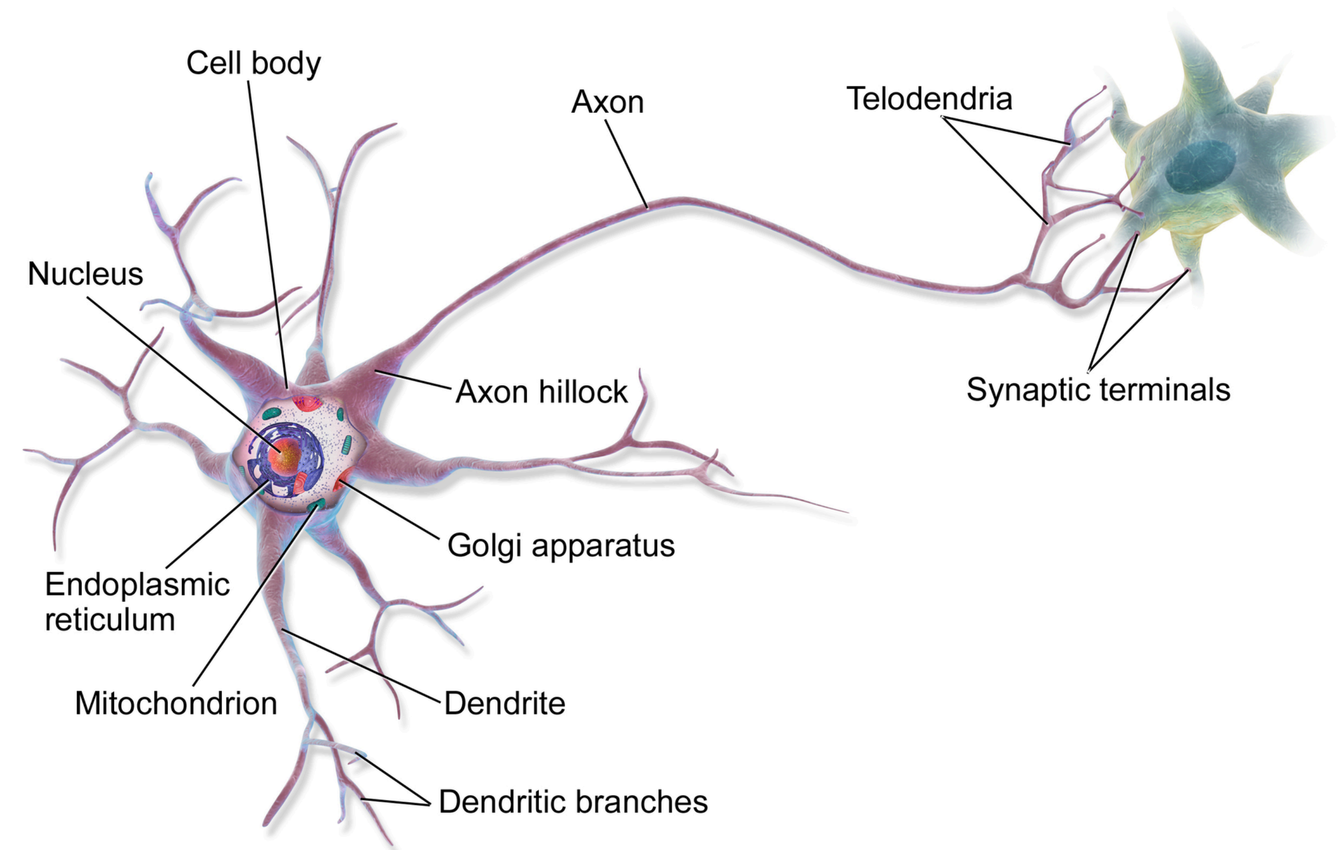


https://en.wikipedia.org/wiki/Neuron#/media/File:Blausen_0657_MultipolarNeuron.png

Biological neural networks

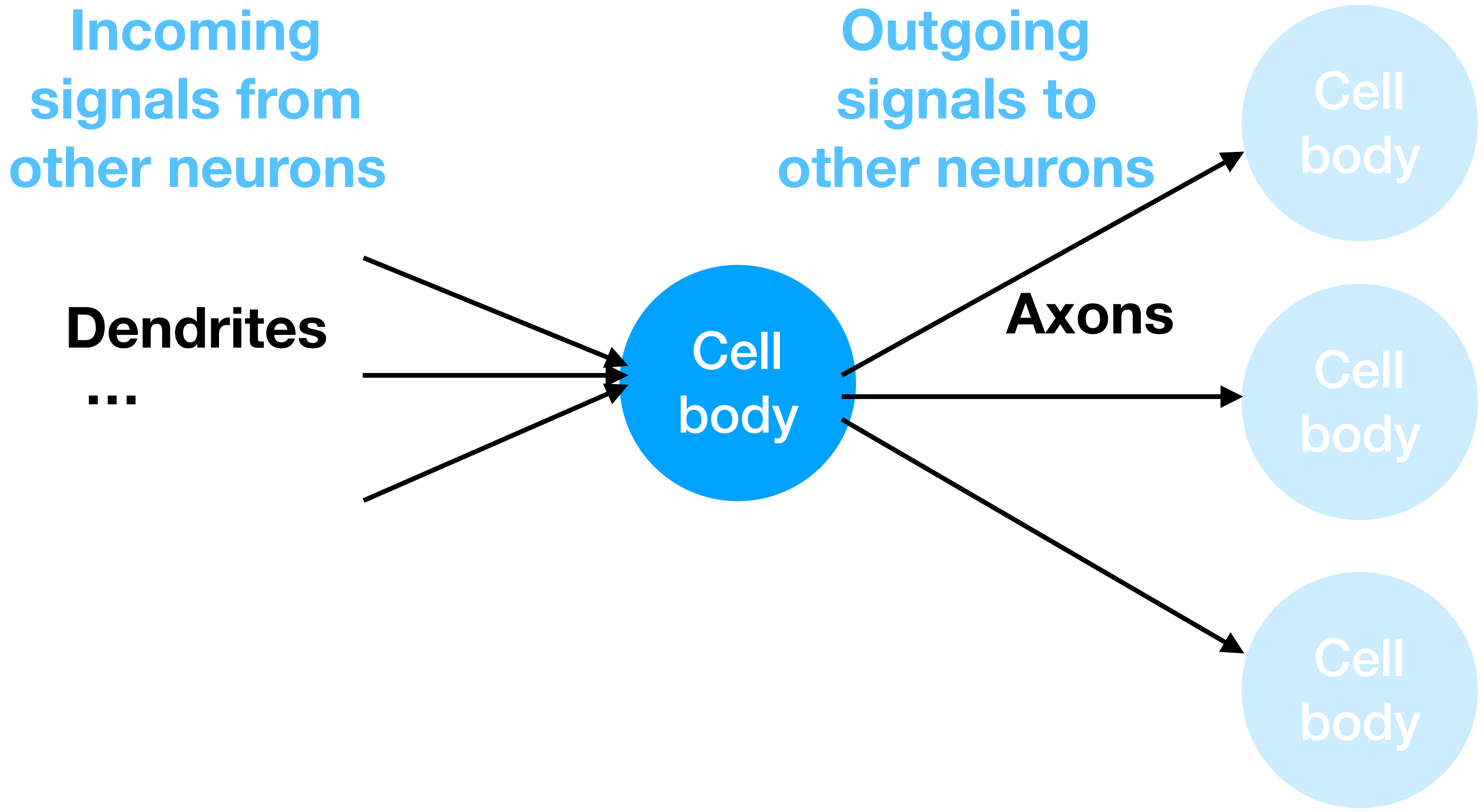
- Neurons can transmit electrical potentials to other neurons via chemical neurotransmitters.

- When the voltage change within one neuron exceeds some threshold, an **all-or-none** electrical potential is transmitted along the axon to neighboring neurons.



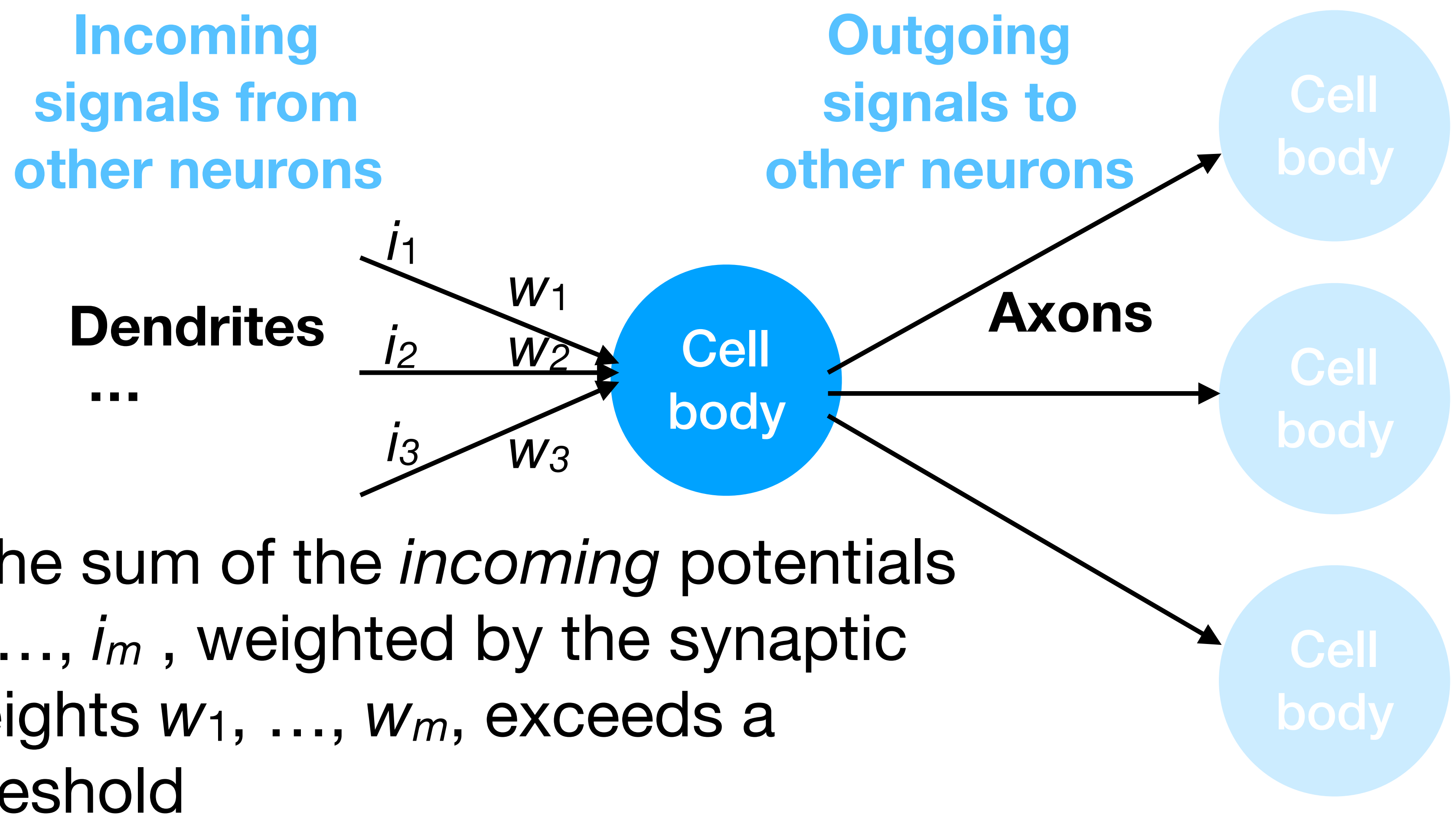
Artificial neural networks

- We can model this process using an artificial neural network:



Artificial neural networks

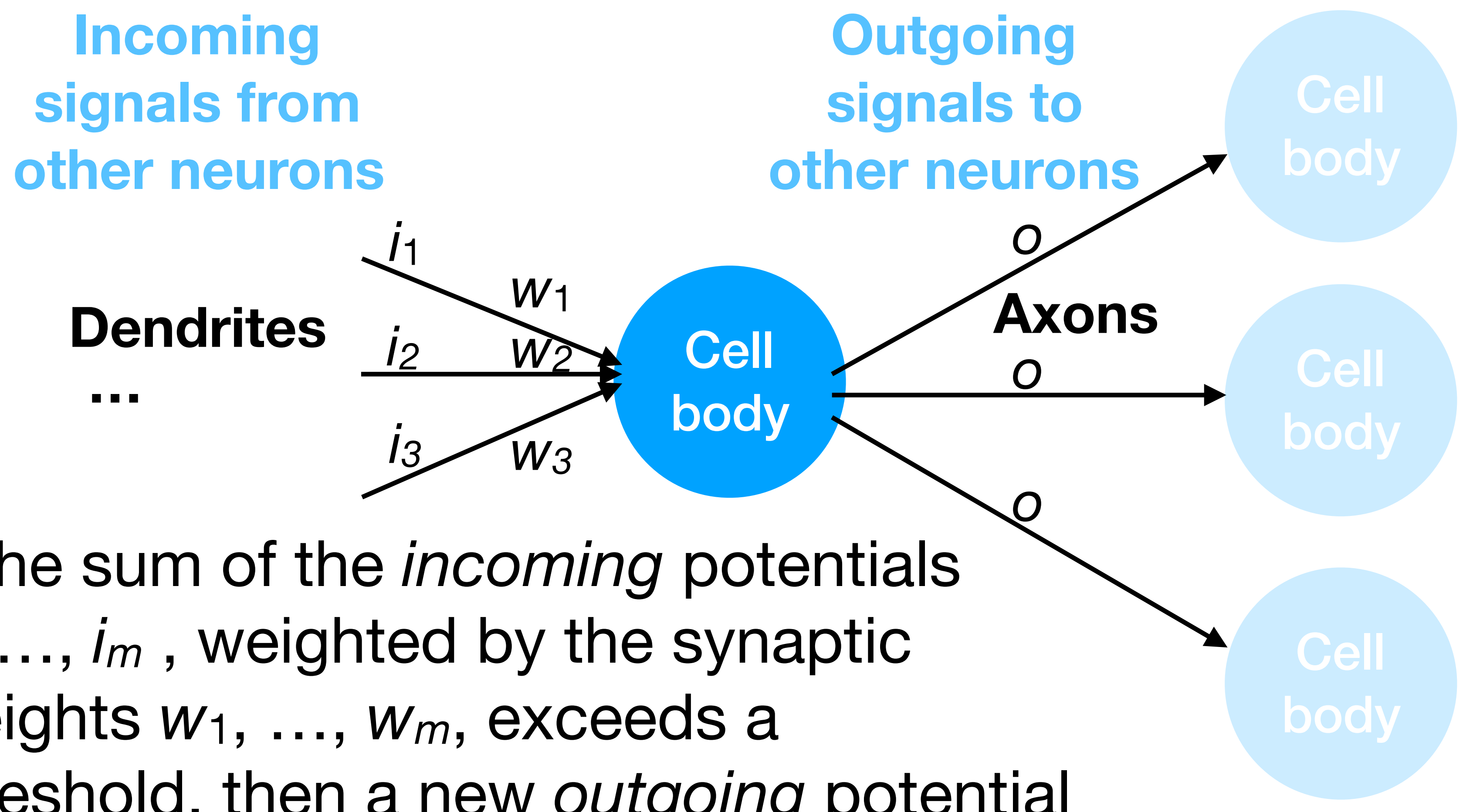
- We can model this process using an artificial neural network:



- If the sum of the *incoming* potentials $i_1, \ldots, i_m$, weighted by the synaptic weights $w_1, \ldots, w_m$, exceeds a threshold

Artificial neural networks

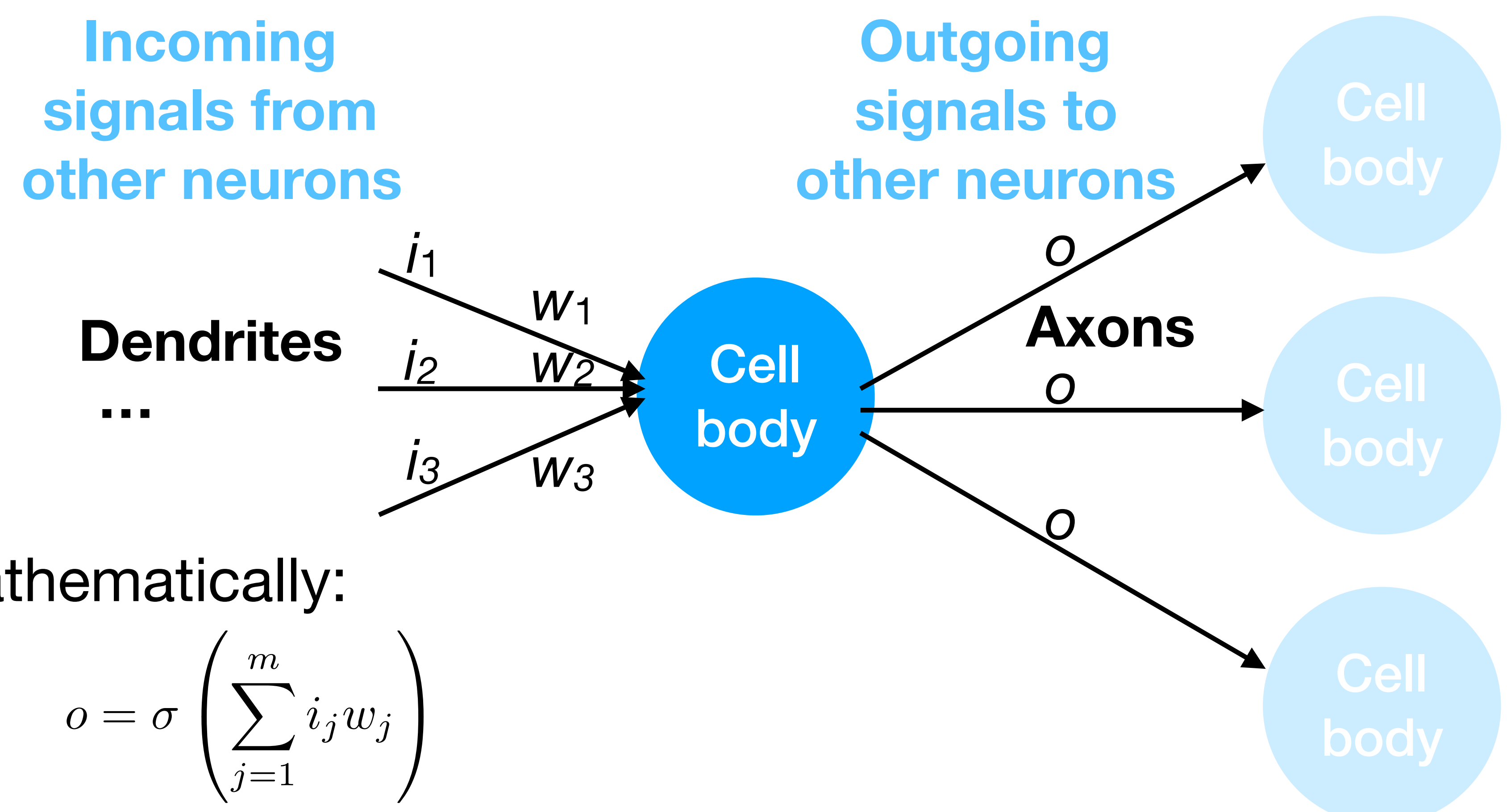
- We can model this process using an artificial neural network:



- If the sum of the *incoming* potentials $i_1, \ldots, i_m$, weighted by the synaptic weights $w_1, \ldots, w_m$, exceeds a threshold, then a new *outgoing* potential o is transmitted along the axons.

Artificial neural networks

- We can model this process using an artificial neural network:



- Mathematically:

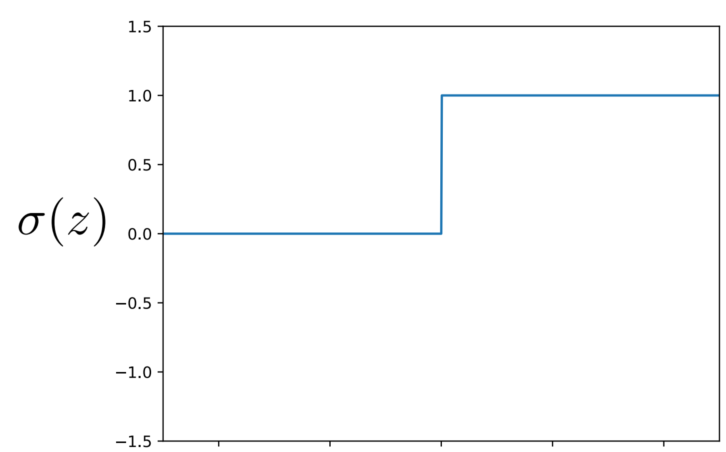
$$o = \sigma\left(\sum_{j=1}^{m} i_j w_j\right)$$

where σ is (usually) some non-linear **activation function.**

Activation functions

- One of the key ingredients of artificial neural networks is the choice of activation function σ.

- In the original Perceptron neural network, (Rosenblatt 1957), σ was the **Heaviside step function**:



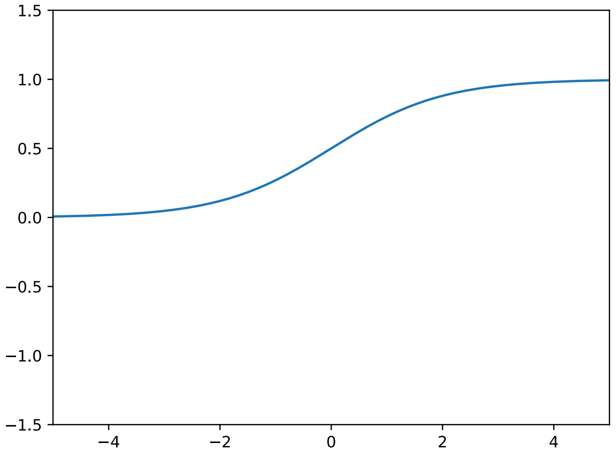
$$z = \sum_{j=1}^{m} i_j w_j$$

which directly models the all-or-none firing rule of biological neural networks.

Activation functions

- Because the Heaviside step function is non-differentiable at 0, it was largely replaced by either a **logistic sigmoid**:

$$\sigma(z) = \frac{1}{1 + \exp -z}$$

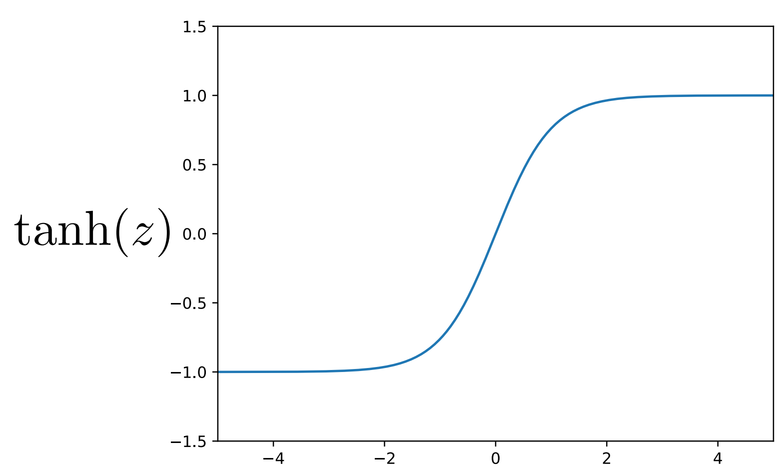


$\sigma(z)$

$$z = \sum_{j=1}^{m} i_j w_j$$

Activation functions

- …or sometimes with hyperbolic tangent:

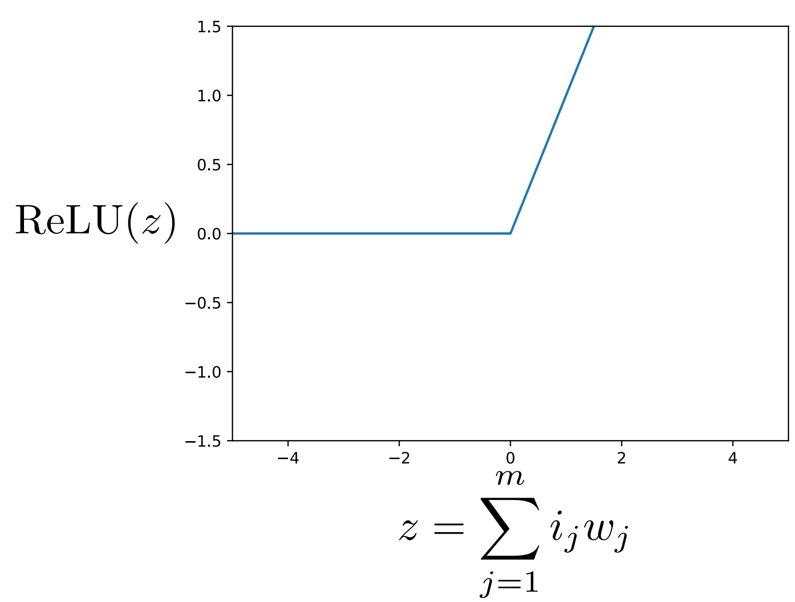


$$z = \sum_{j=1}^{m} i_j w_j$$

Activation functions

- More modern neural networks often use a **rectified linear** activation function, known as **ReLU**:

$$\mathrm{ReLU}(z) = \max\{0, z\}$$

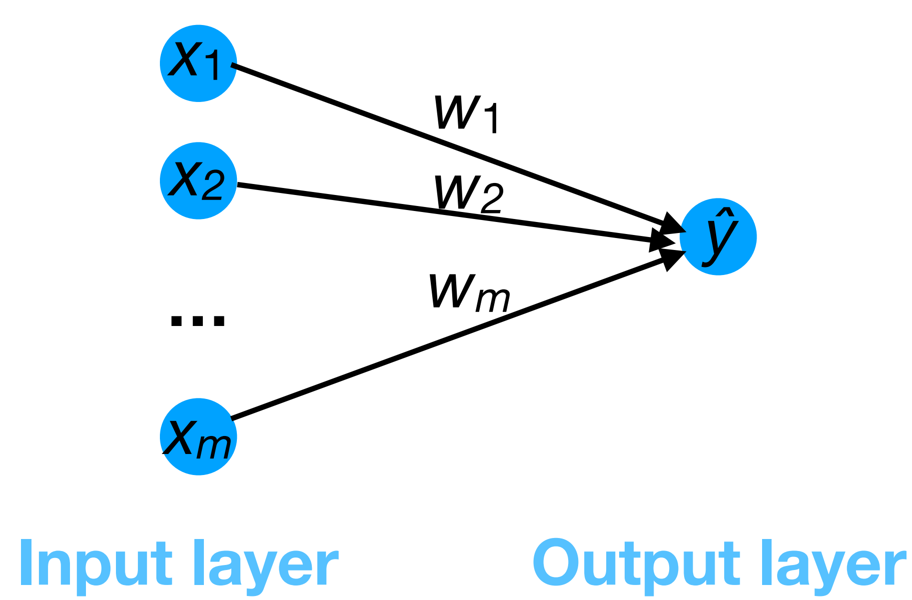


NNs as a mathematical function

- Neural networks can compute mathematical functions if we designate a subset of neurons as the input and a subset of neurons as the output.

- One common network design is a **feed-forward** (FF) network, consisting of multiple layers of neurons, each of which feeds to the next layer.

- Here is a simple example, which is equivalent to linear regression:

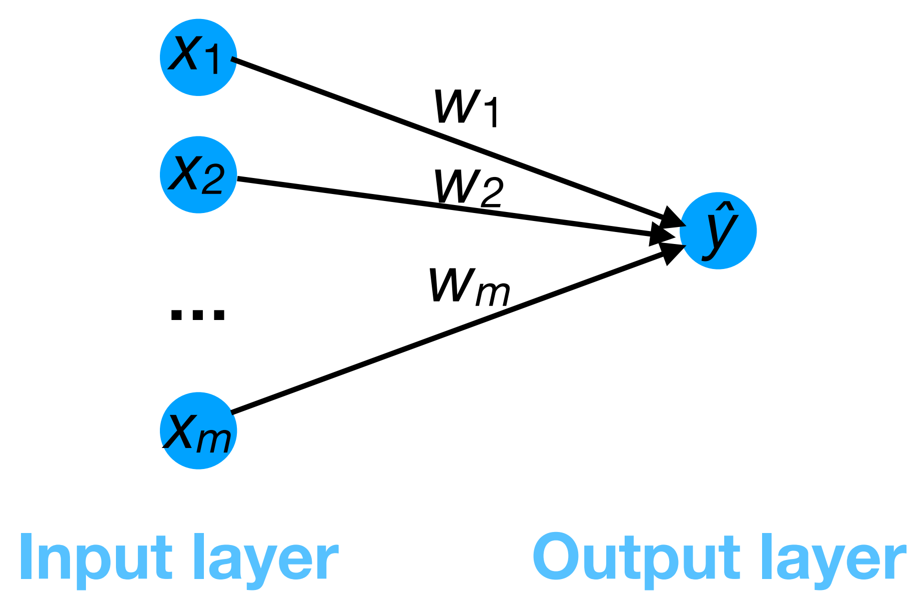


NNs as a mathematical function

- The input layer **x** directly transmits the values $x_1, \ldots, x_m$ to the next layer.

- The output layer $\hat{y}$ computes the sum of the inputs multiplied by the synaptic weights **w**.

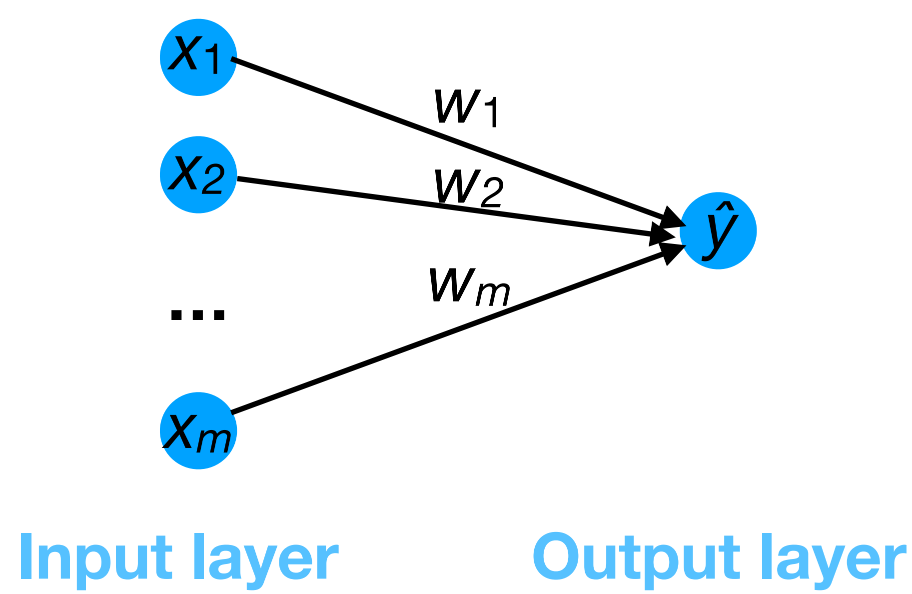
- In this network, no activation function was used to transform the weighted sum of incoming potentials to $\hat{y}$.



NNs as a mathematical function

- This network computes the function:

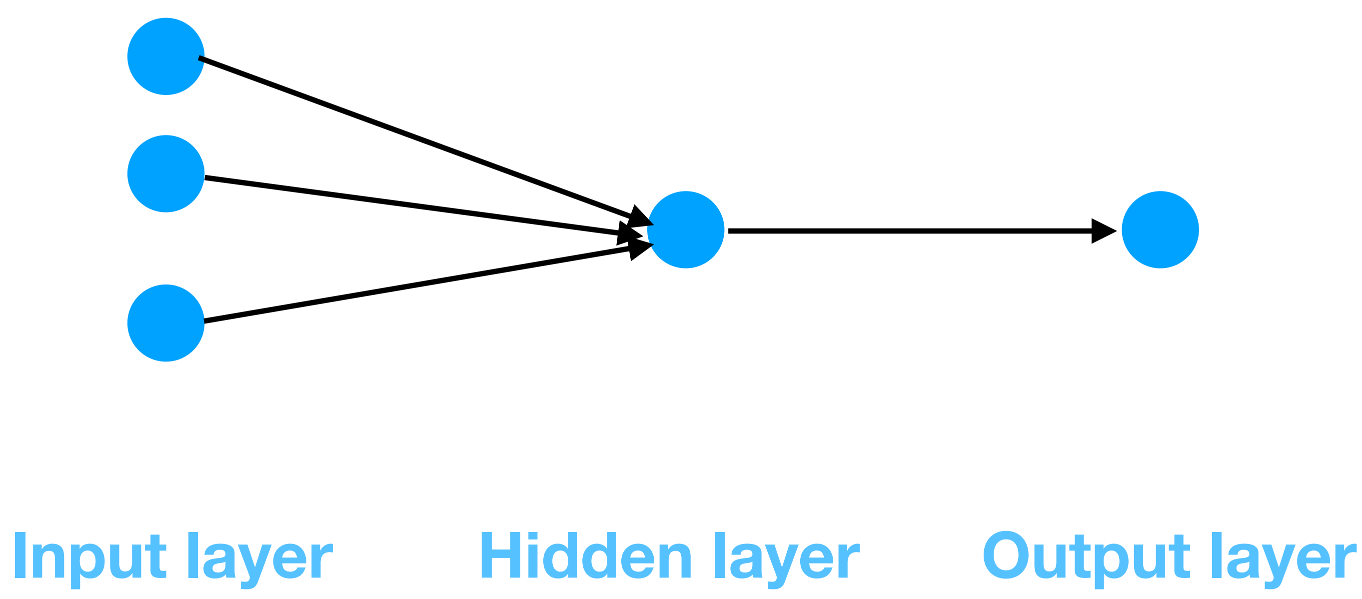
$$\hat{y} = f(\mathbf{x}) = \sum_{j=1}^{m} \mathbf{x}_j \mathbf{w}_j$$



Example: feed-forward NN

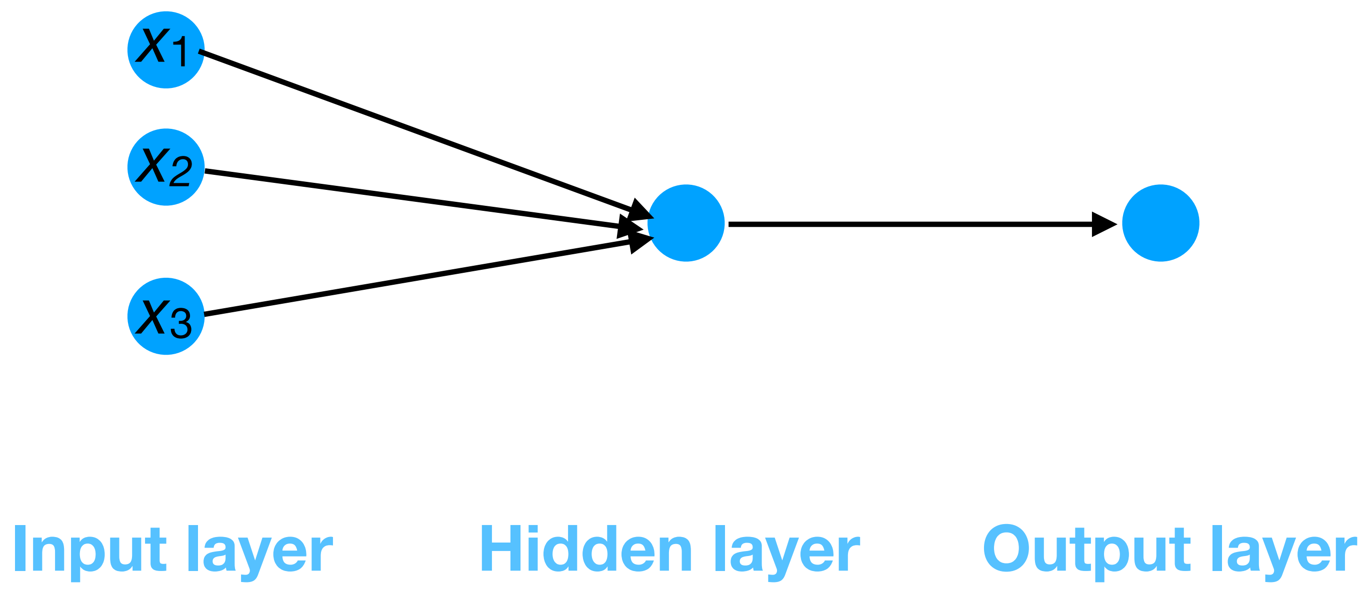
- More commonly, there is at least one layer between the input and output layers; it is known as a **hidden layer**.

- The NN processes the input according to the direction of the arrows…



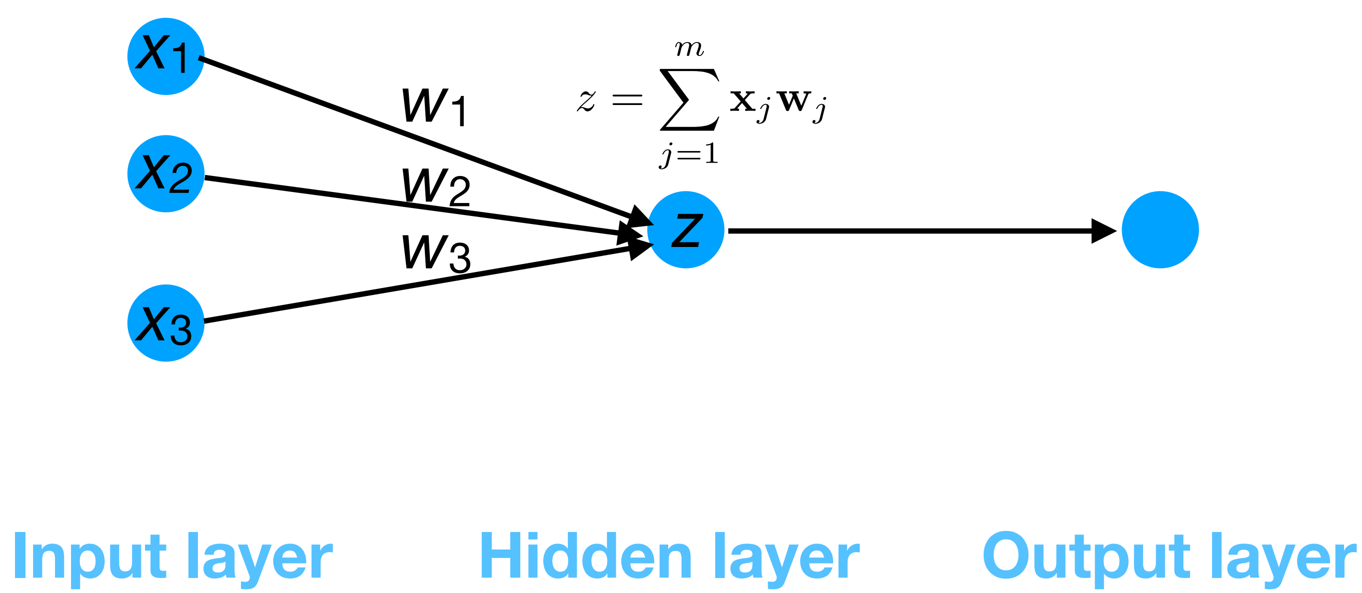
Example: feed-forward NN

- First, we just plug in the input **x** into the first layer:



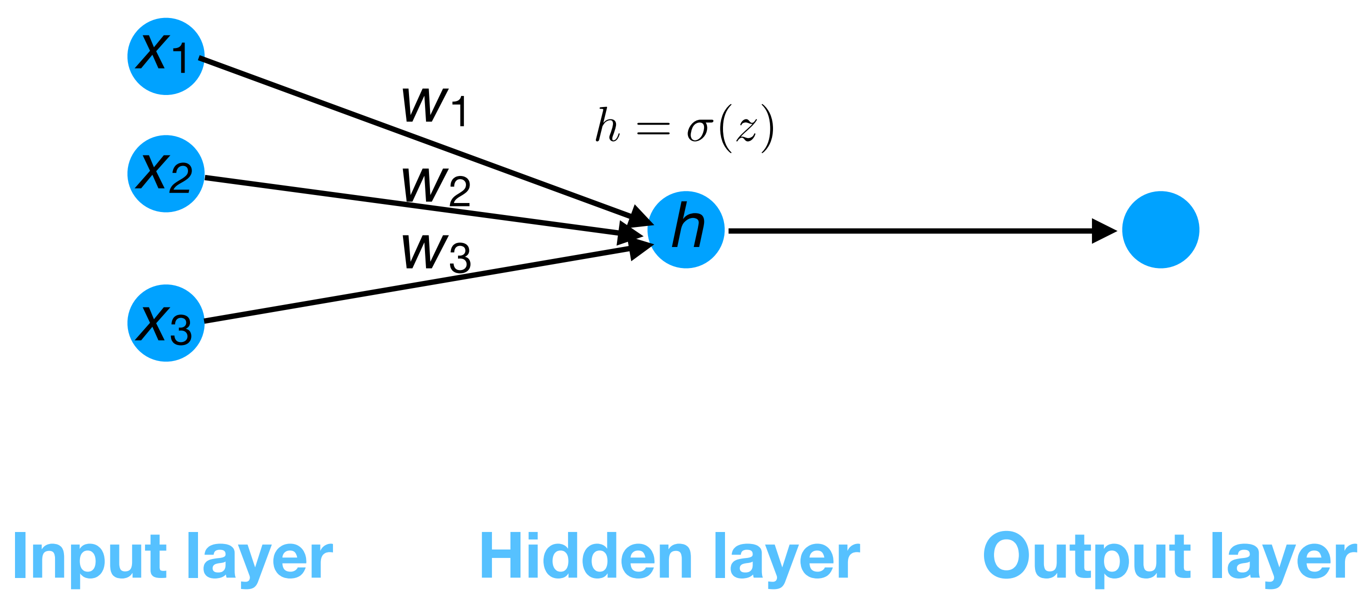
Example: feed-forward NN

- Next, we compute the sum of incoming potentials to the neuron in the hidden layer:



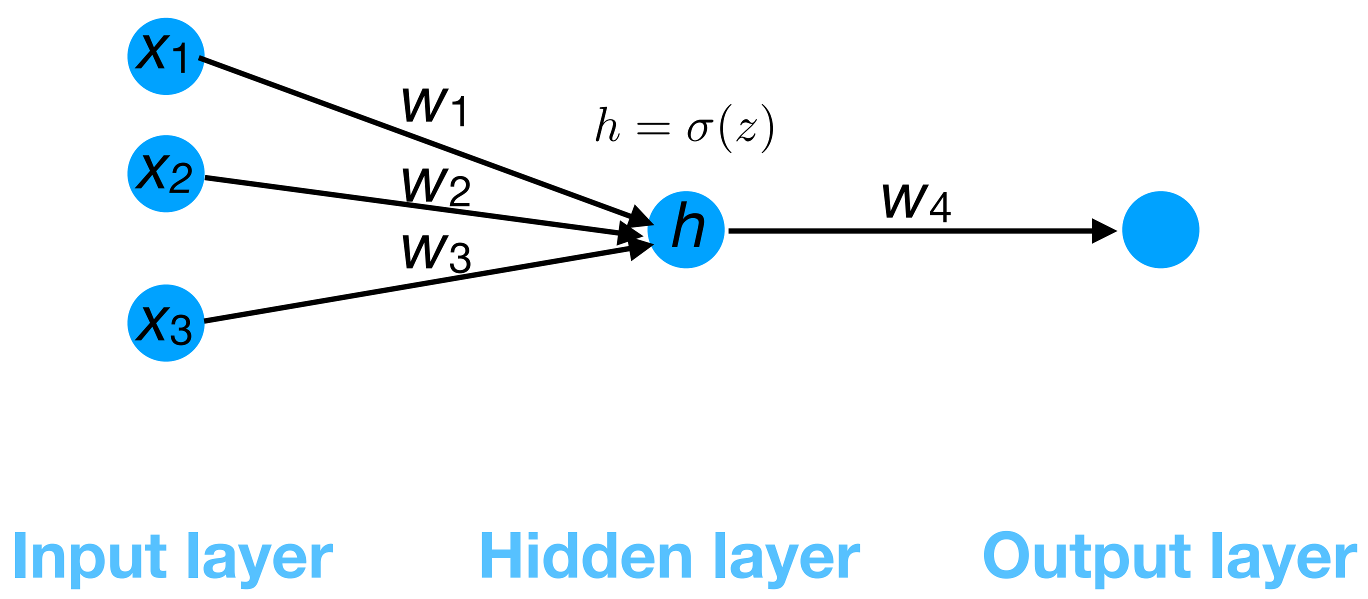
Example: feed-forward NN

- We then pass the z to the activation function σ (which could be the logistic sigmoid, tanh, ReLU, etc.) to get h:



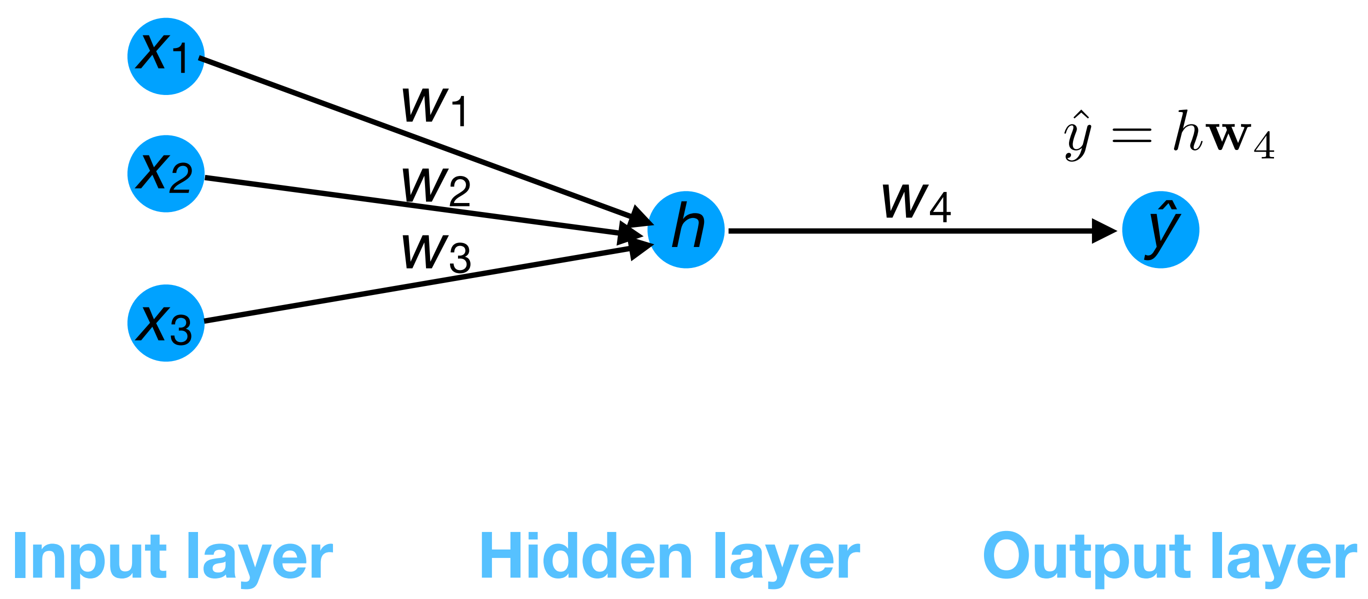
Example: feed-forward NN

- Continuing on, we transmit *h* to the output layer…



Example: feed-forward NN

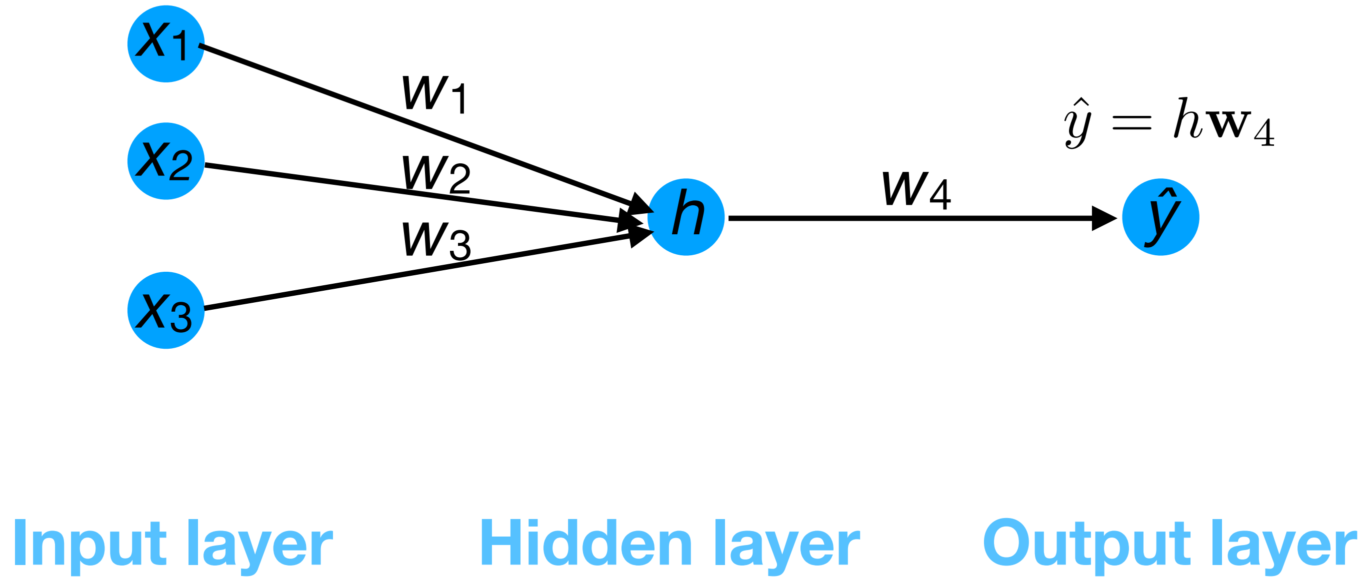
- …to obtain the output $\hat{y}$:



Example: feed-forward NN

- In aggregate, our NN below computes the function:

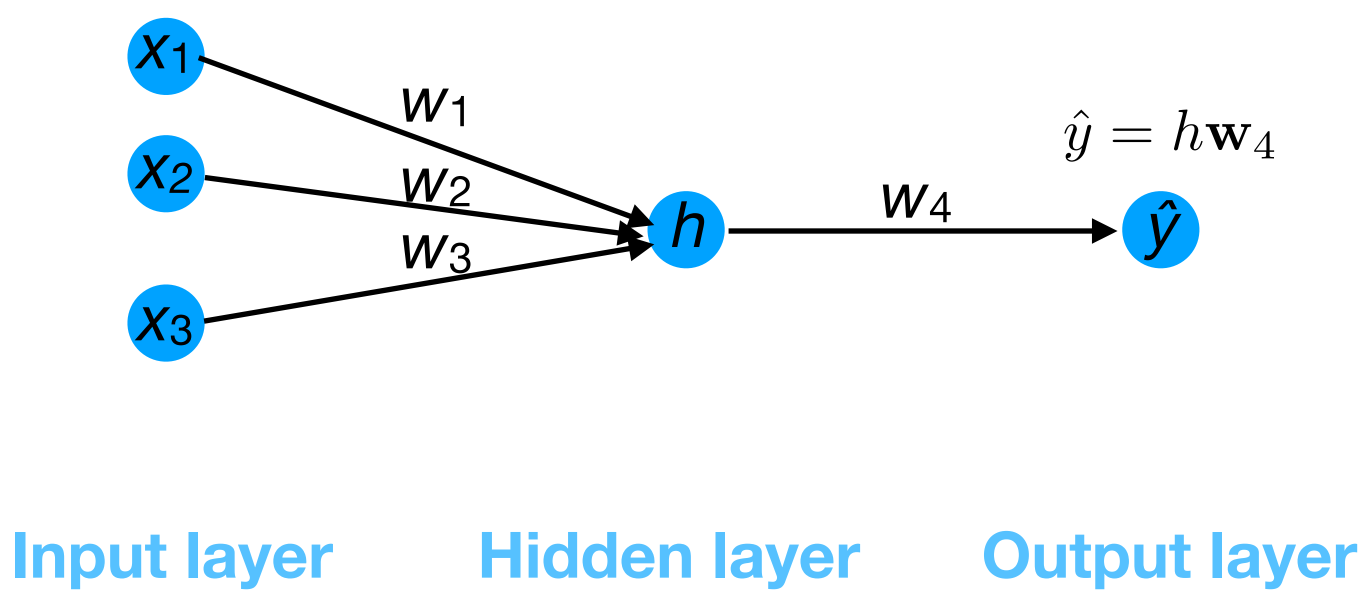
$$\hat{y} = f(\mathbf{x}) = \mathbf{w}_4 \sigma \left(\sum_{j=1}^{3} \mathbf{x}_j \mathbf{w}_j \right)$$



Exercise: feed-forward NN

- What will $\hat{y}$ be for $\mathbf{x}$ = [1 0 -2]^T, $\mathbf{w}_1$=1, $\mathbf{w}_2$=2, $\mathbf{w}_3$=-1.5, and $\mathbf{w}_4$=-1? Assume ReLU is the activation function.

$$\hat{y} = f(\mathbf{x}) = \mathbf{w}_4 \sigma \left(\sum_{j=1}^{3} \mathbf{x}_j \mathbf{w}_j \right)$$



Exercise: feed-forward NN

- What will $\hat{y}$ be for $\mathbf{x}$ = [1 0 -2]^T, $\mathbf{w}_1$=1, $\mathbf{w}_2$=2, $\mathbf{w}_3$=-1.5, and $\mathbf{w}_4$=-1? Assume ReLU is the activation function.

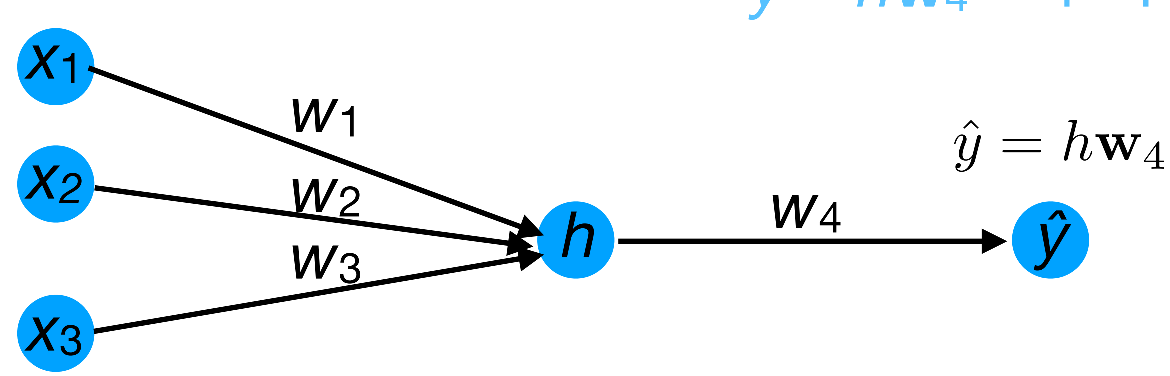
$$\hat{y} = f(\mathbf{x}) = \mathbf{w}_4 \sigma \left(\sum_{j=1}^{3} \mathbf{x}_j \mathbf{w}_j \right)$$

$\mathbf{x}_1\mathbf{w}_1 = 1$, $\mathbf{x}_2\mathbf{w}_2 = 0$, $\mathbf{x}_3\mathbf{w}_3 = 3$

$z = 4$

$h = \text{ReLU}(z) = 4$

$\hat{y} = h\mathbf{w}_4 = 4 * -1 = -4$



Input layer **Hidden layer** **Output layer**