CS 4342: Class 18

Jacob Whitehill

Neural networks

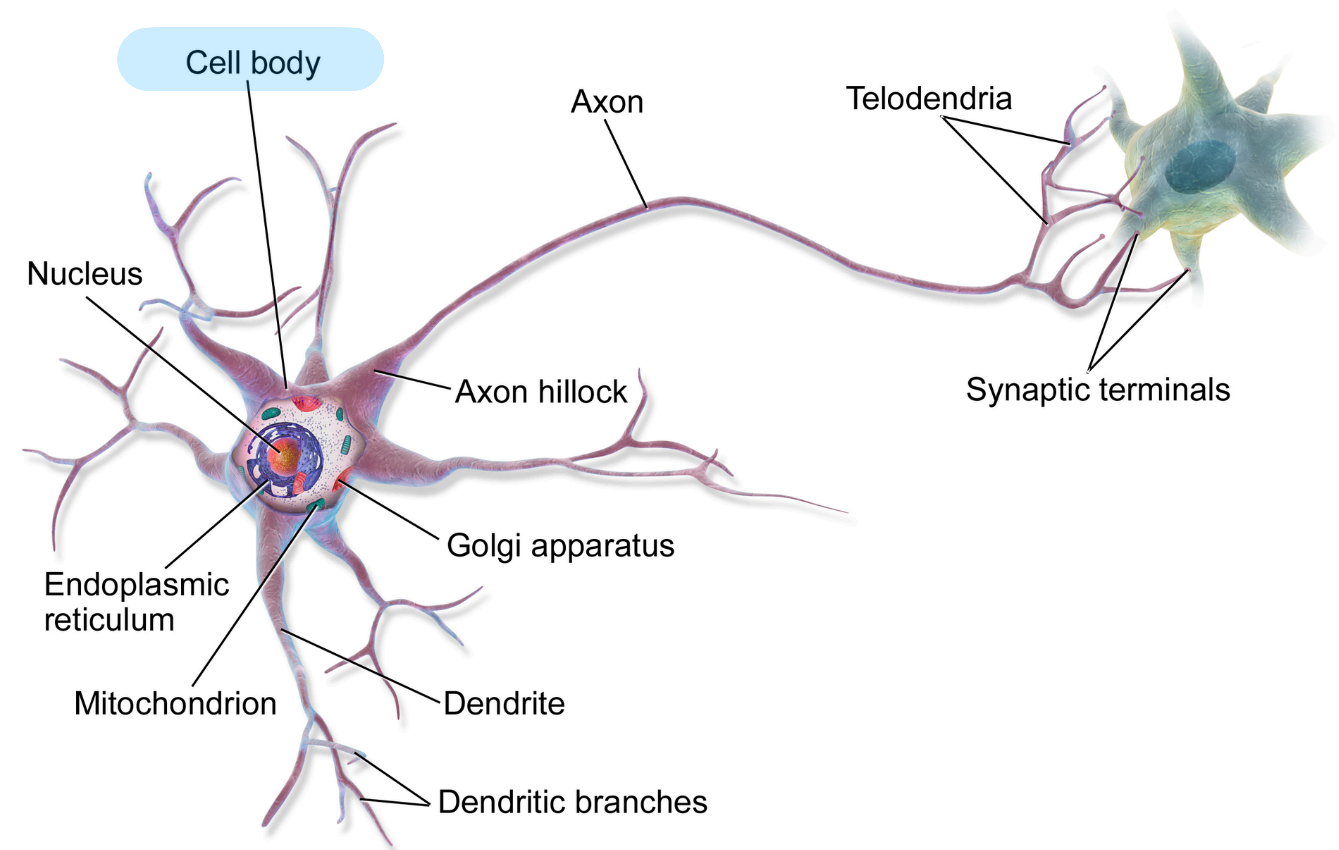
Neural networks

- 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)

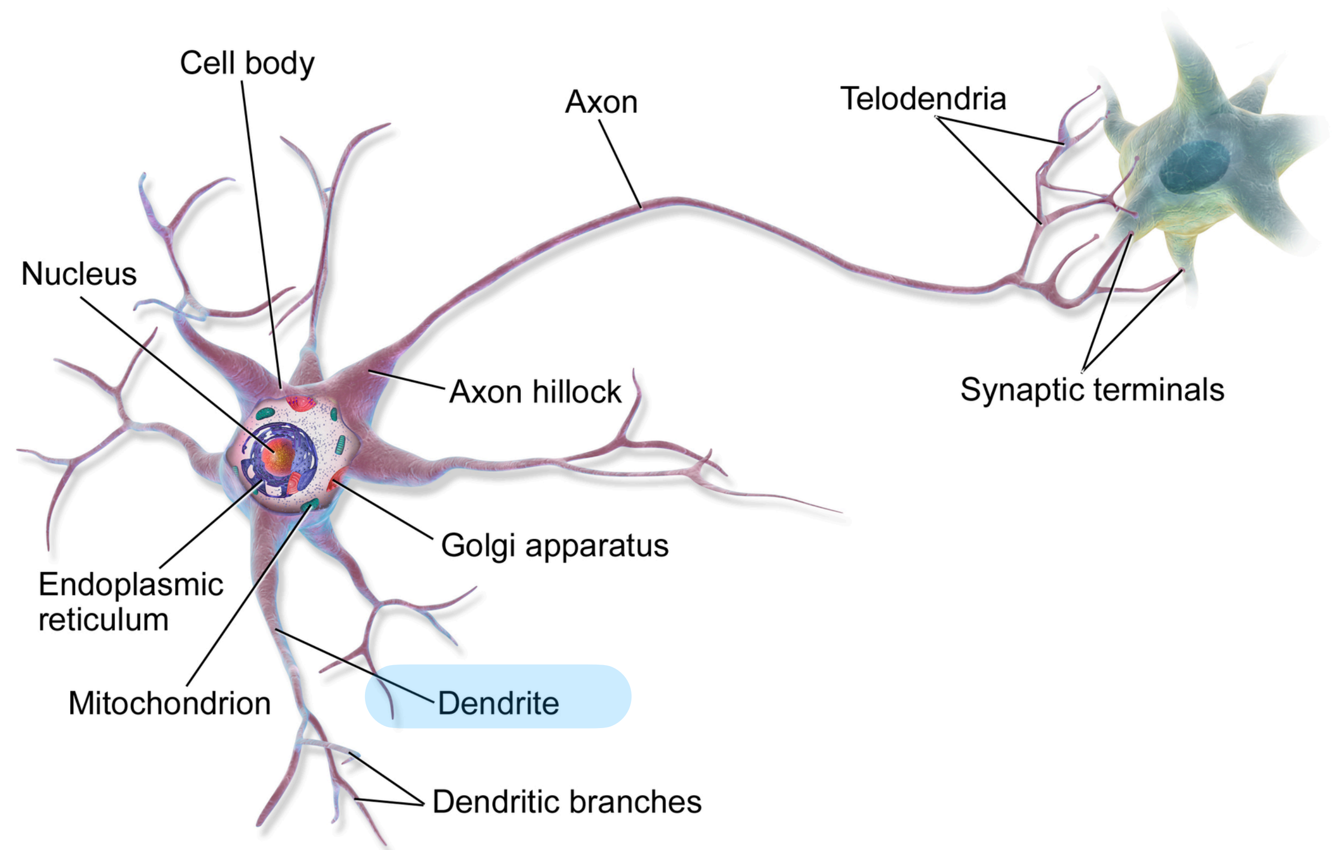


https://en.wikipedia.org/wiki/Neuron#/media/File:Blausen_0657_MultipolarNeuron.png

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)

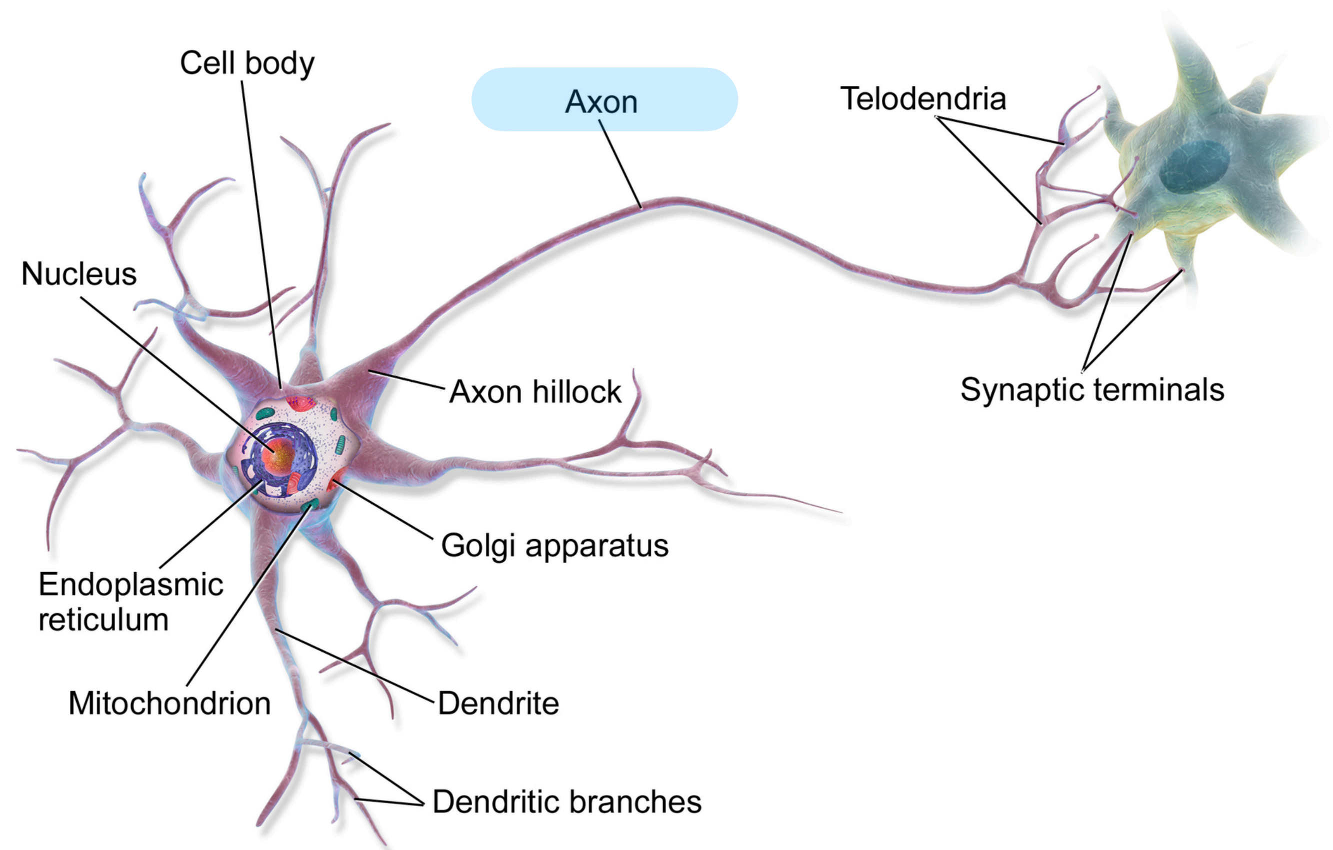


https://en.wikipedia.org/wiki/Neuron#/media/File:Blausen_0657_MultipolarNeuron.png

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)

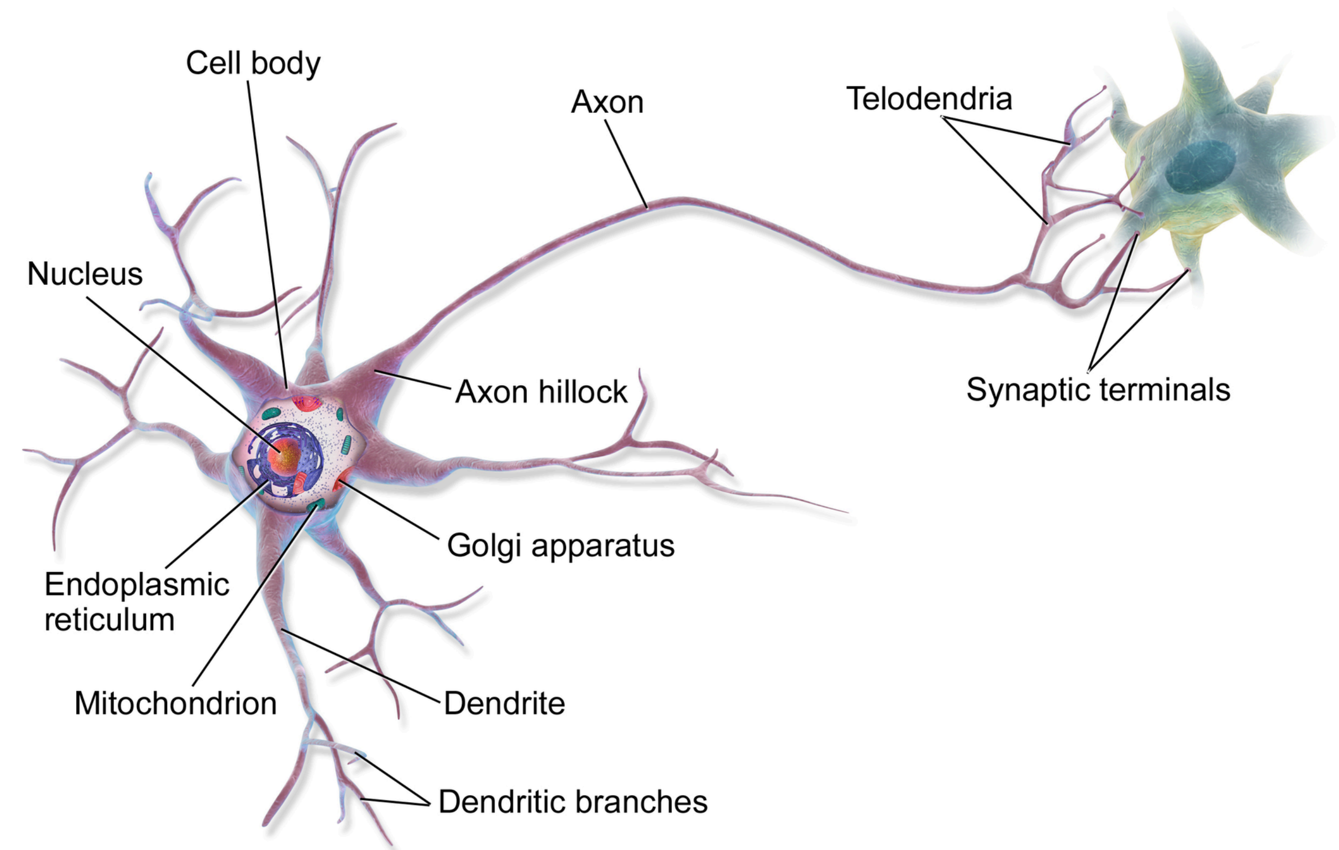


https://en.wikipedia.org/wiki/Neuron#/media/File:Blausen_0657_MultipolarNeuron.png

Biological neural networks

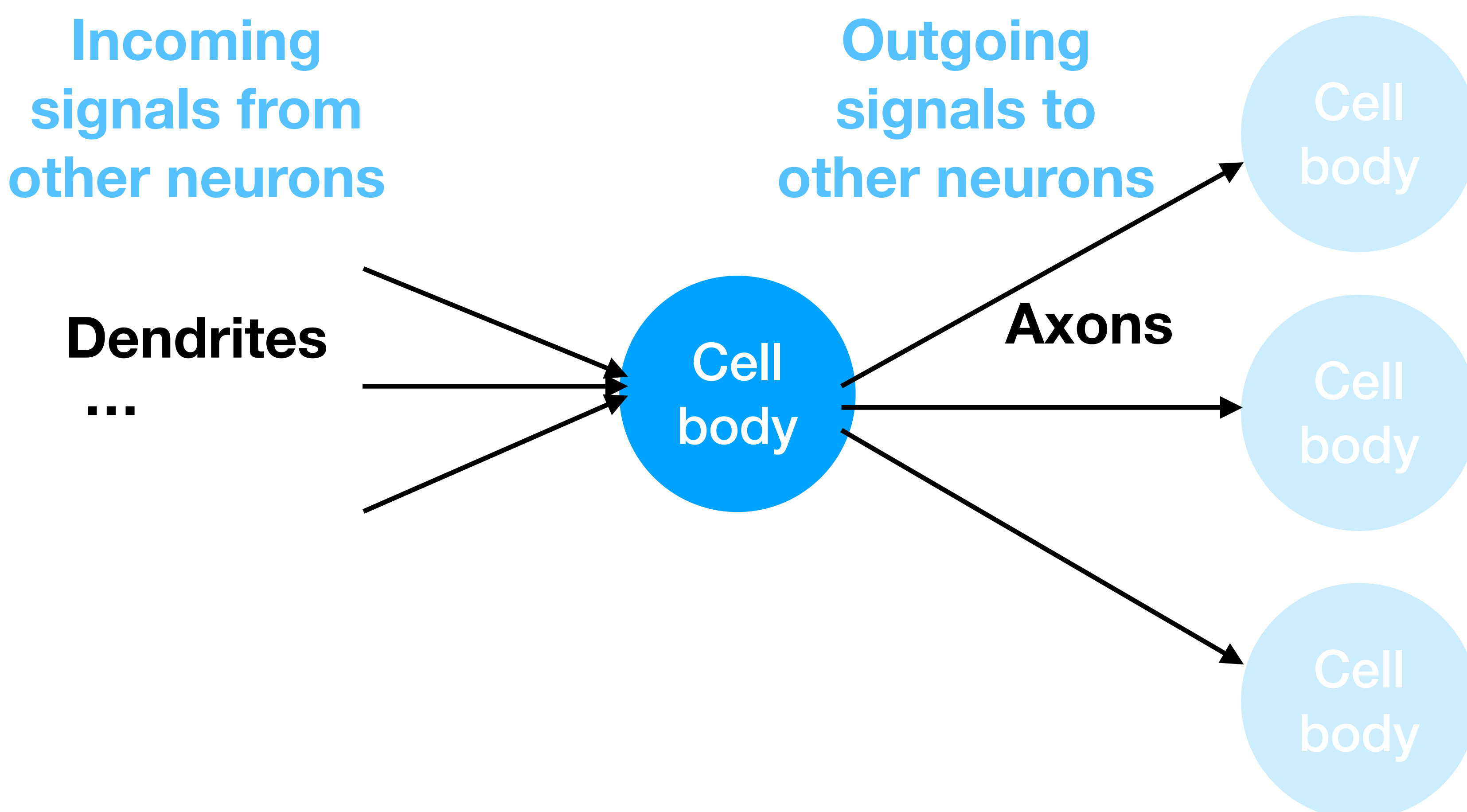
- Neurons can transmit electrical potentials to other neurons via chemical neurotransmitters.

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



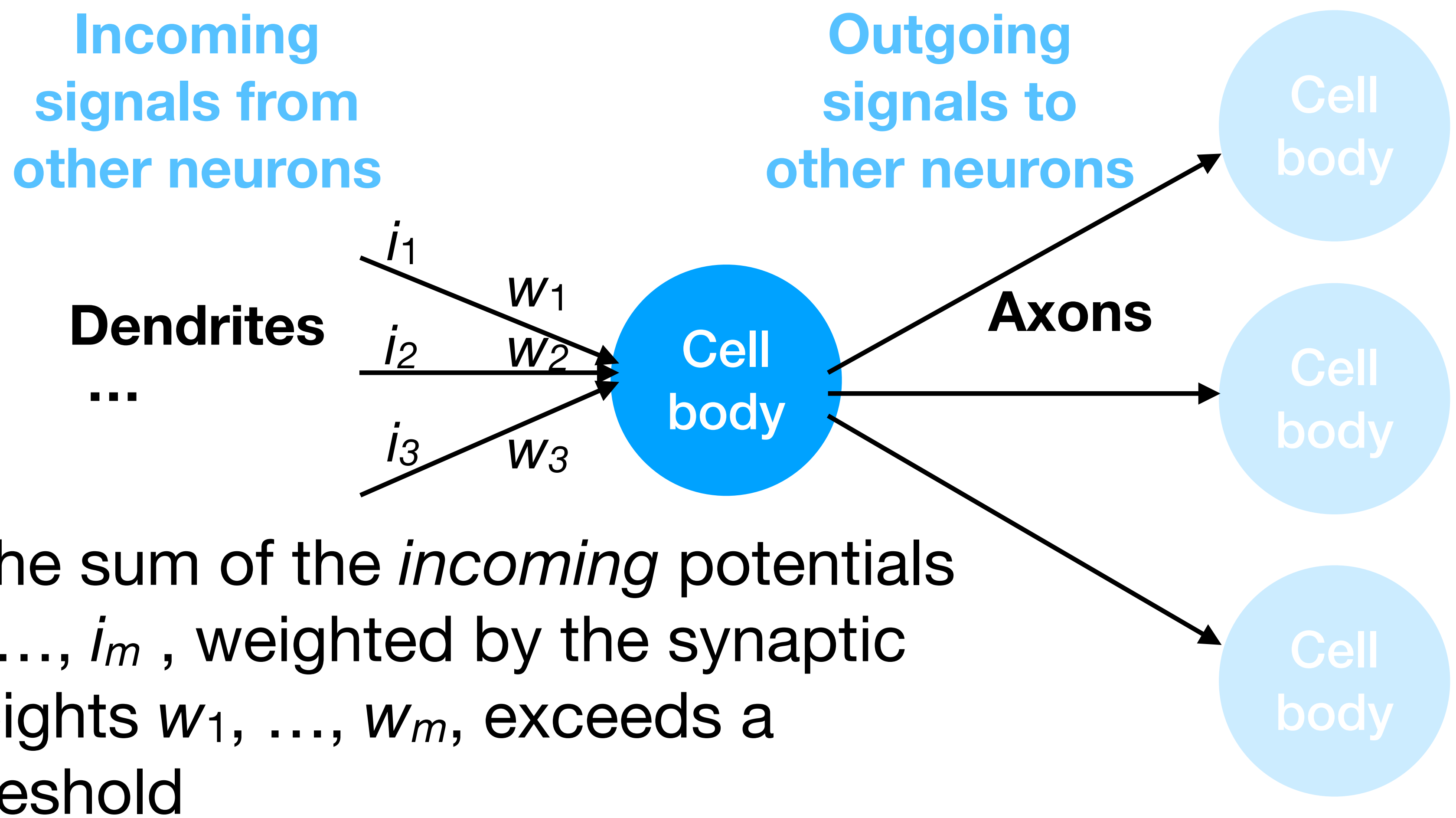
Artificial neural networks

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



Artificial neural networks

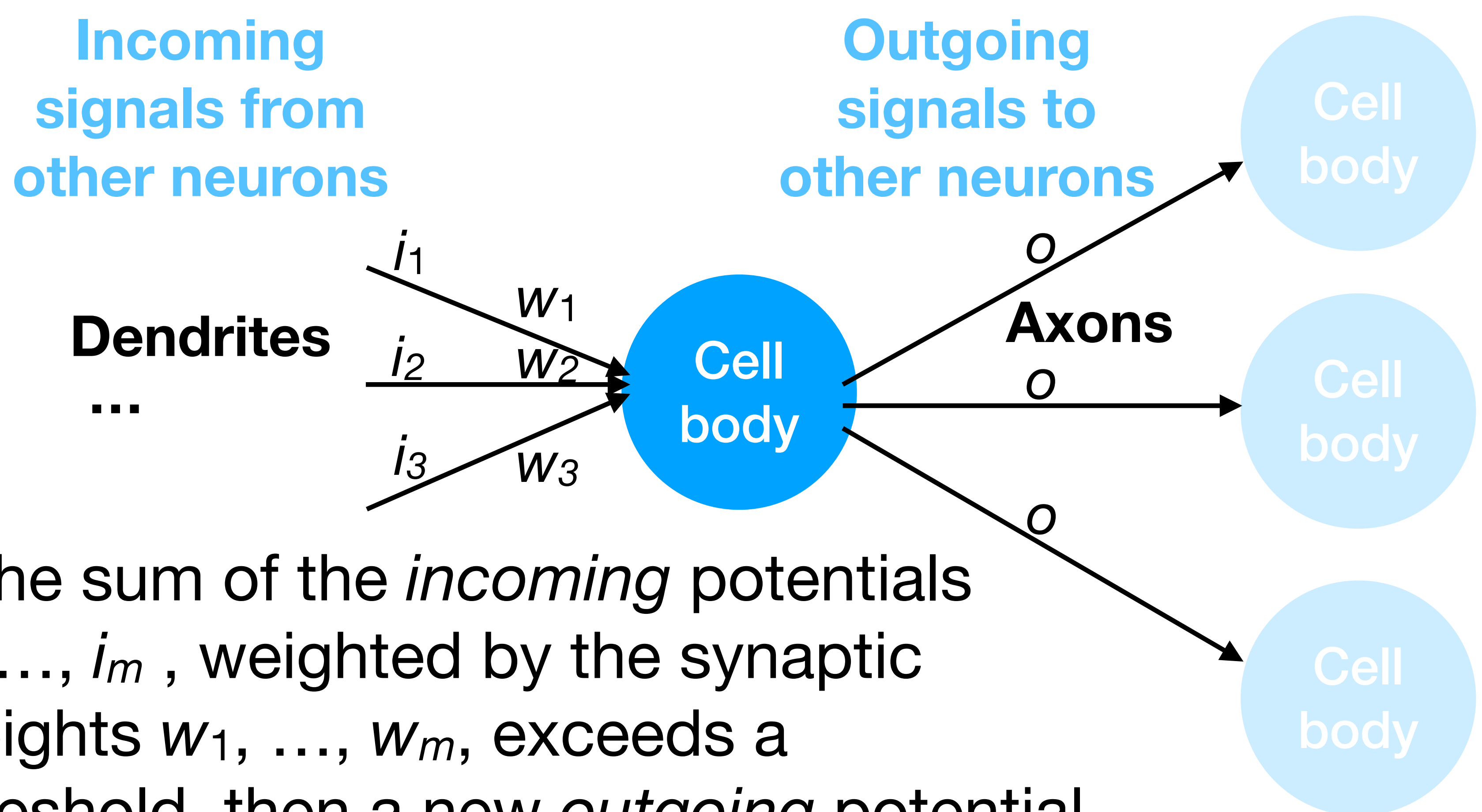
- We can model this process using an artificial neural network:



- 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

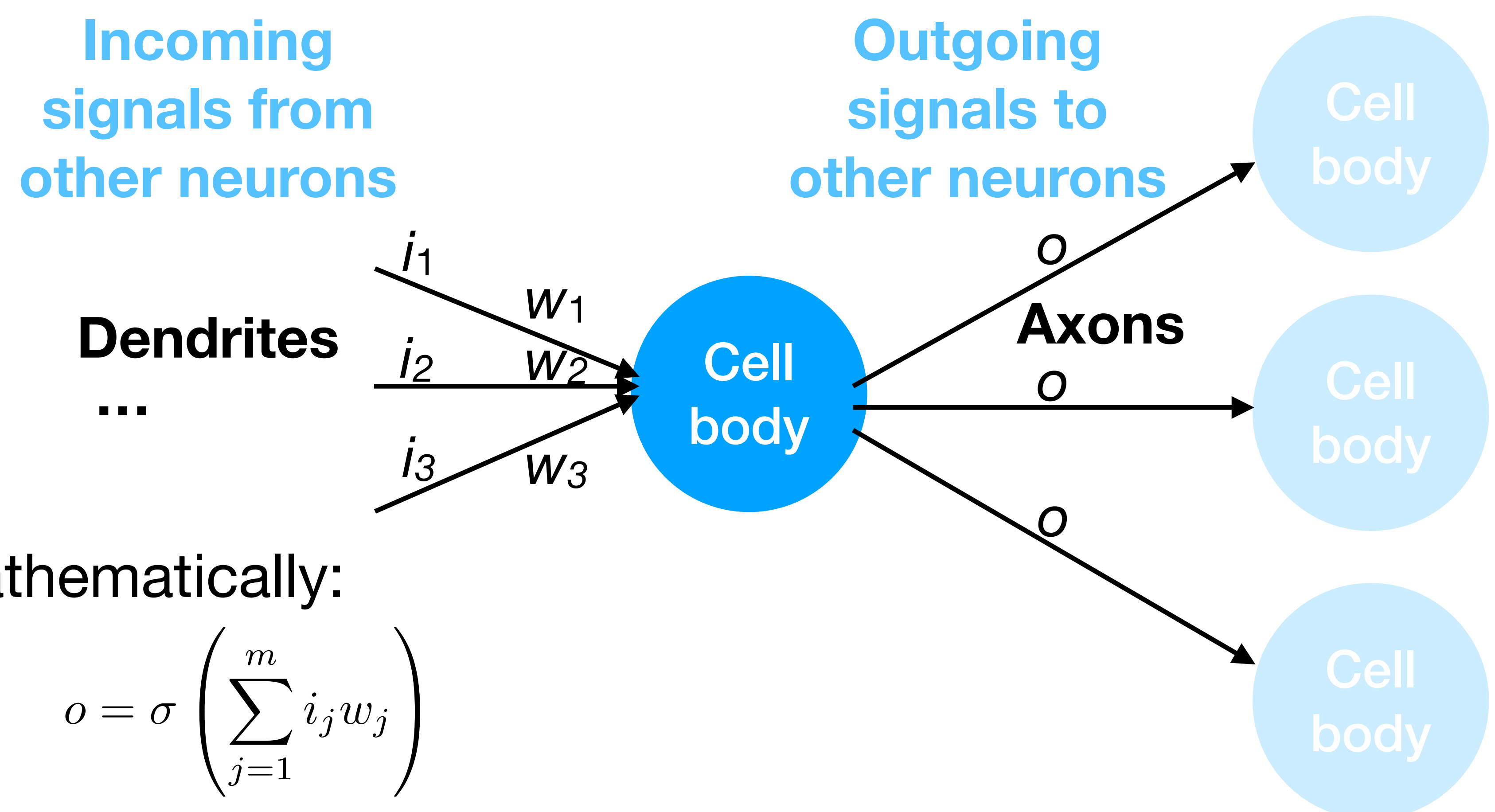
- We can model this process using an artificial neural network:



- 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:

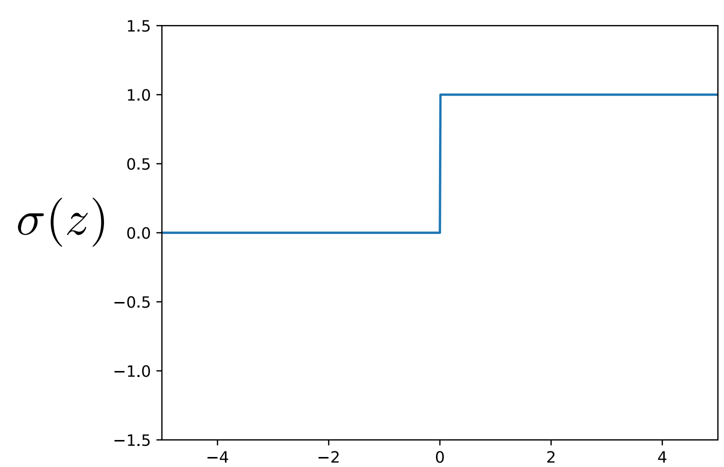
$$o = \sigma\left(\sum_{j=1}^{m} i_j w_j\right)$$

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**:



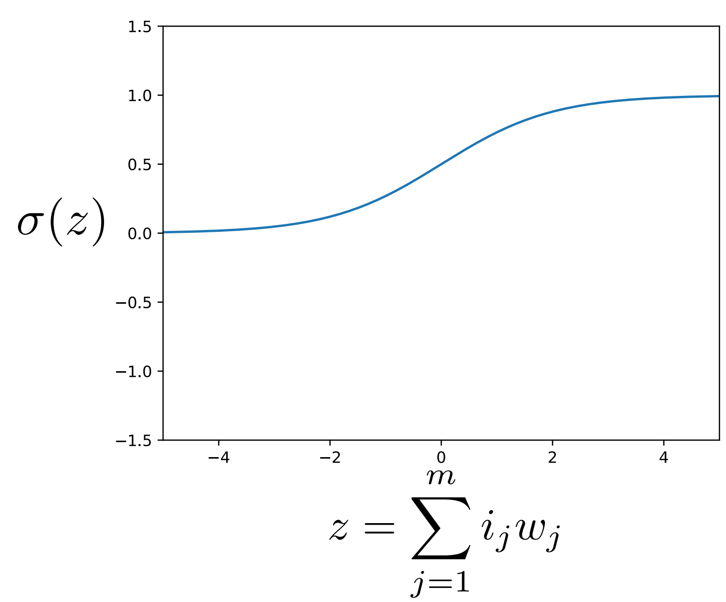
$$z = \sum_{j=1}^{m} i_j w_j$$

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

Activation functions

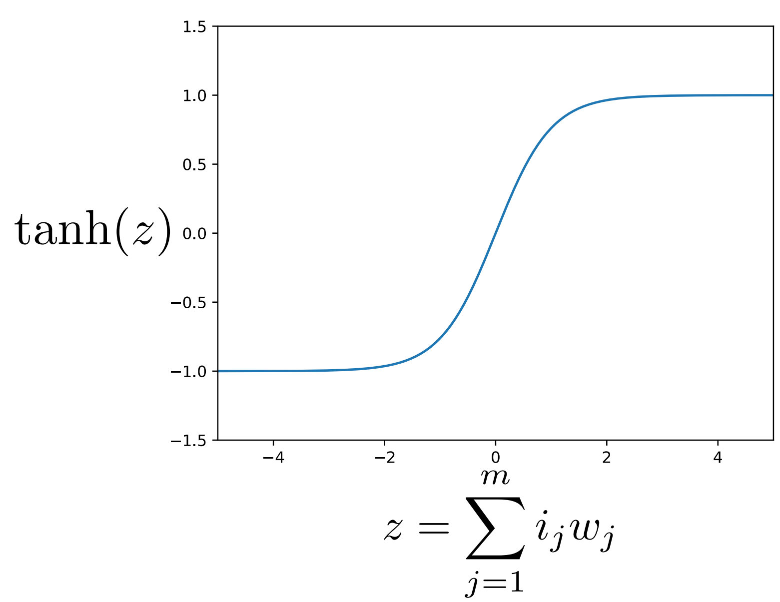
- Because the Heaviside step function has 0 gradient almost everywhere, it was largely replaced by either a **logistic sigmoid**:

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



Activation functions

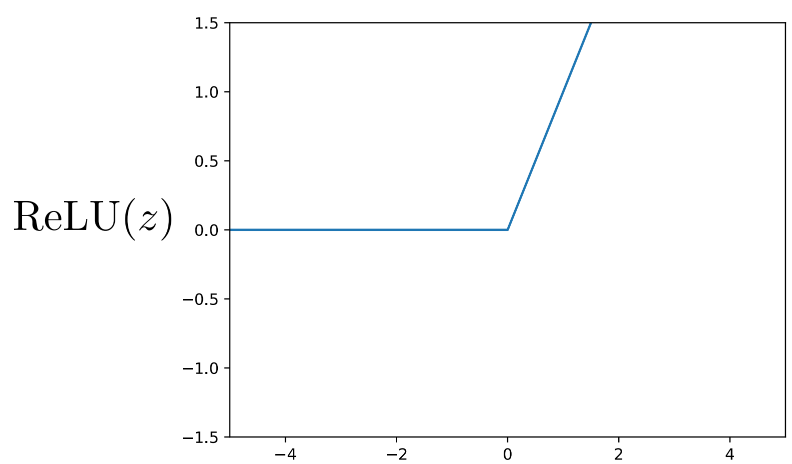
- …or sometimes with hyperbolic tangent:



Activation functions

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

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



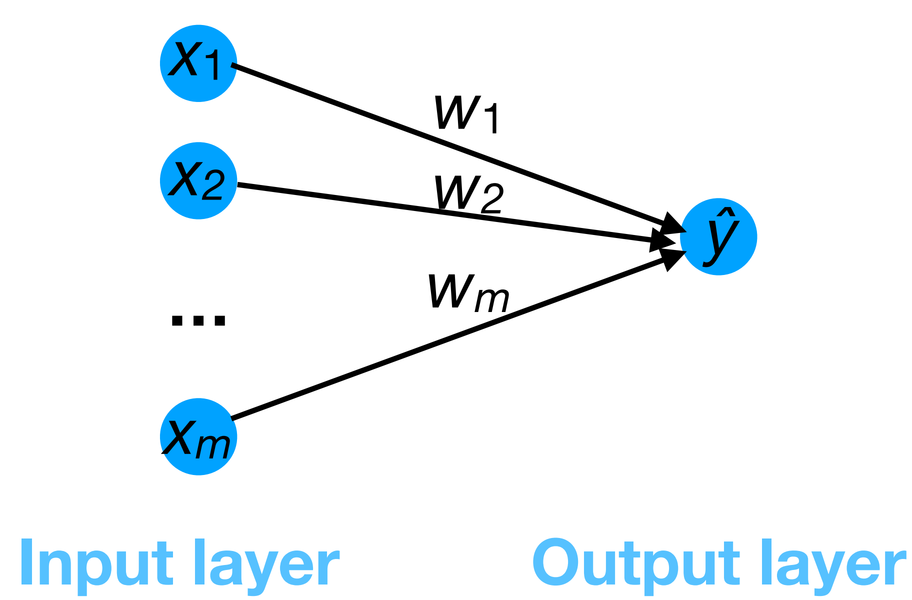
$$z = \sum_{j=1}^{m} i_j w_j$$

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.

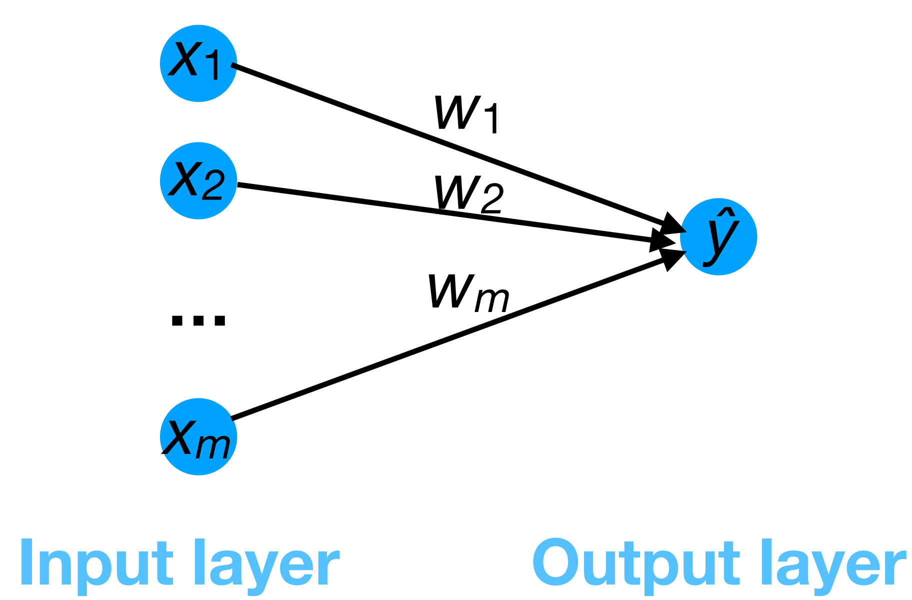
- One common network design is a **feed-forward** (FF) network, consisting of multiple layers of neurons, each of which feeds to the next layer.

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



NNs as a mathematical function

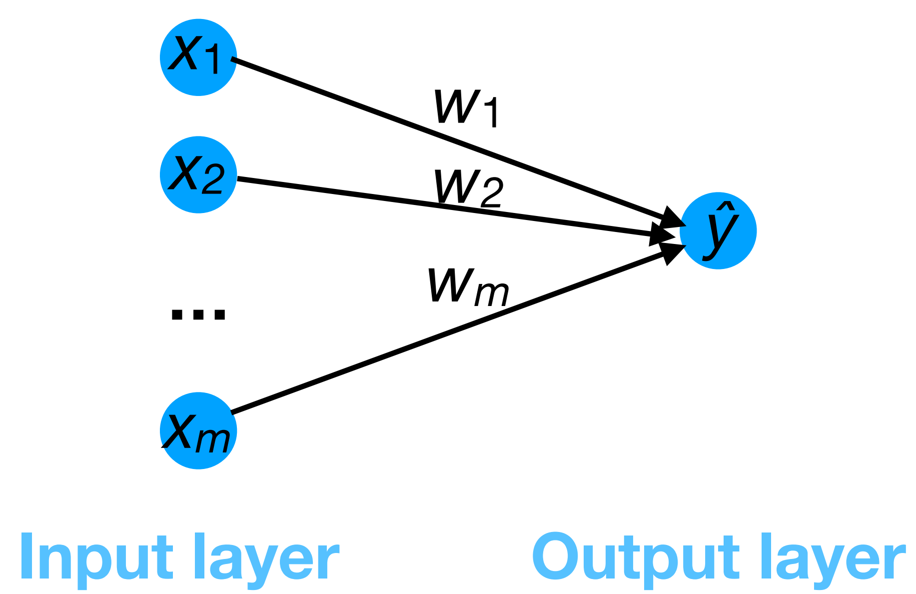
- The input layer **x** directly transmits the values $x_1, \ldots, x_m$ to the next layer.
- The output layer $\hat{y}$ computes the sum of the inputs multiplied by the synaptic weights **w**.
- In this network, no activation function was used to transform the weighted sum of incoming potentials to $\hat{y}$.



NNs as a mathematical function

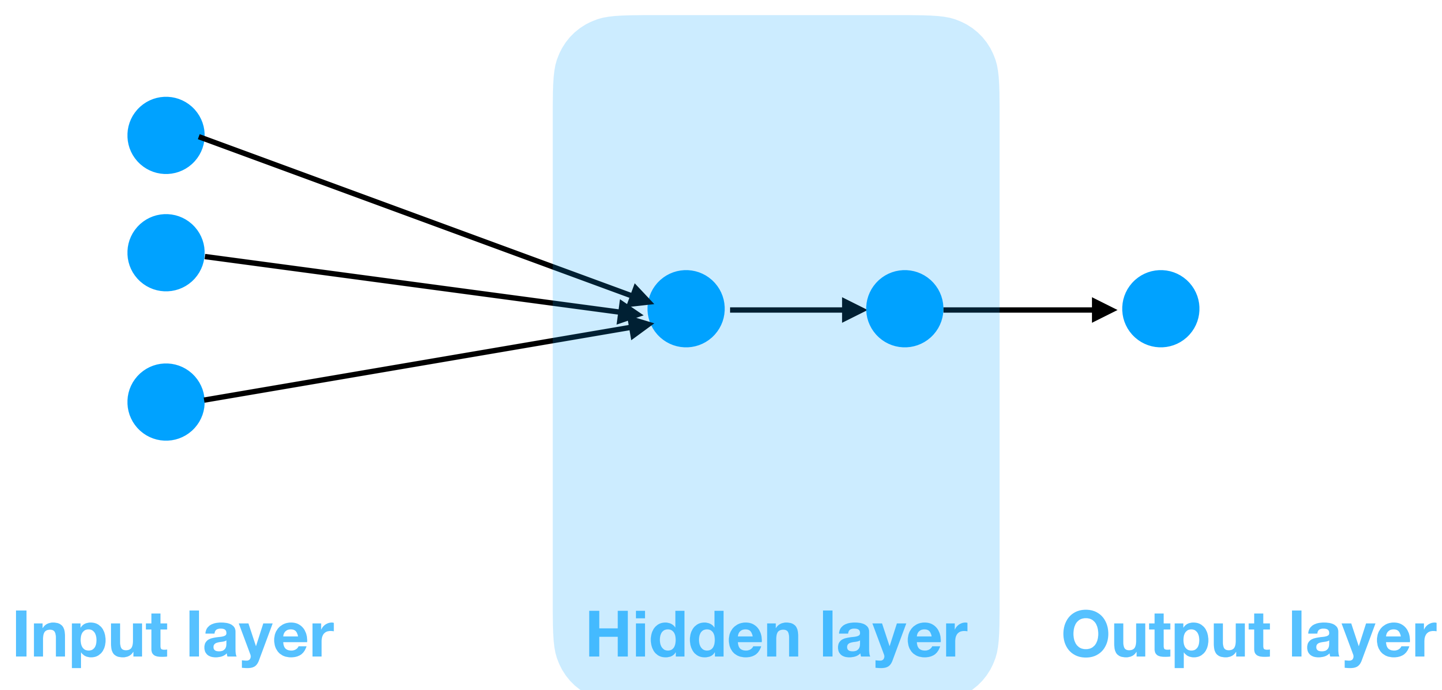
- This network computes the function:

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



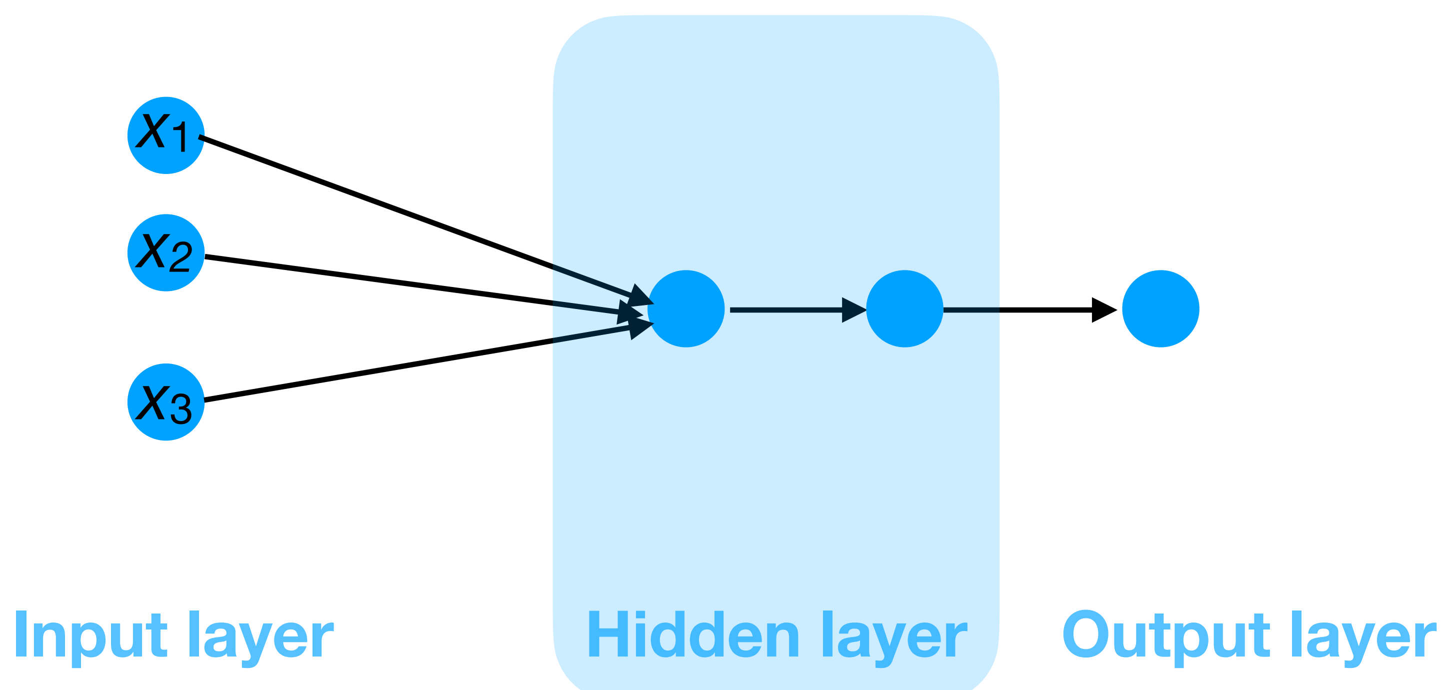
Example: feed-forward NN

- 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…



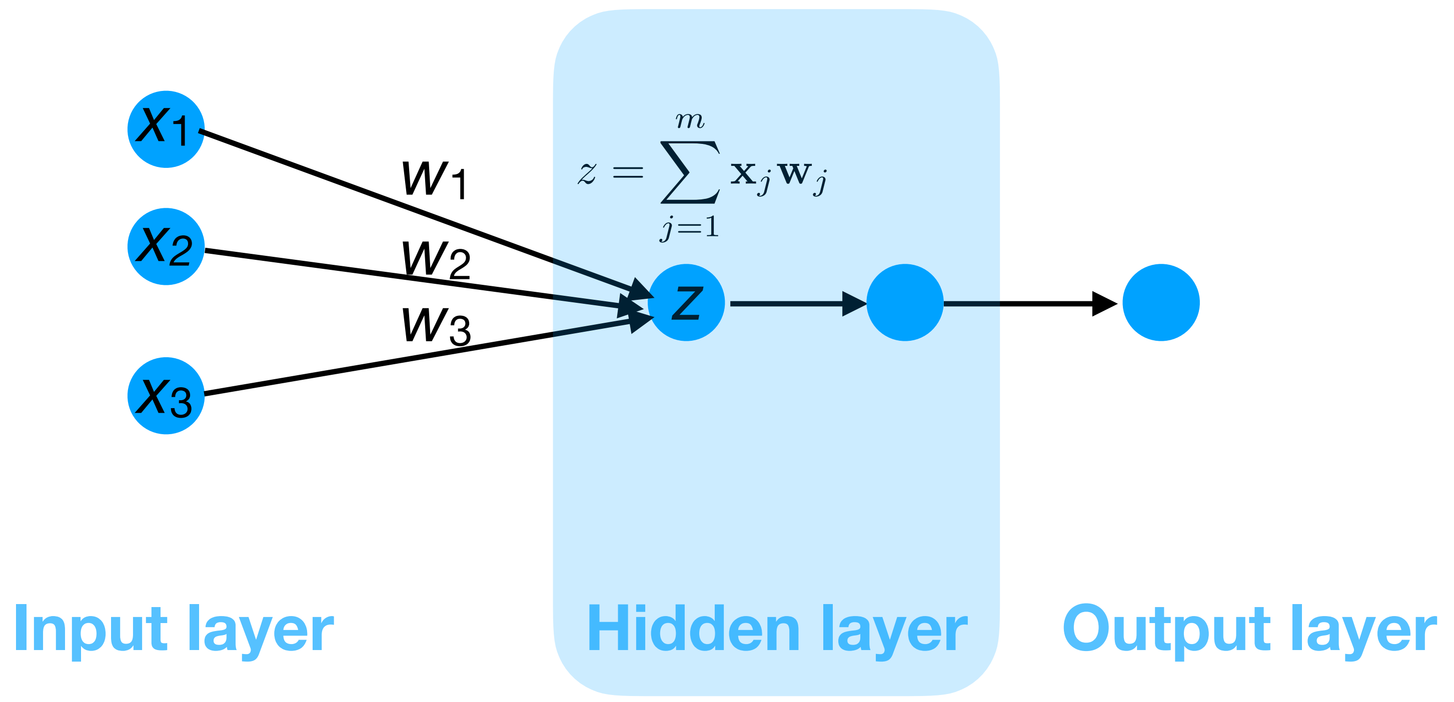
Example: feed-forward NN

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



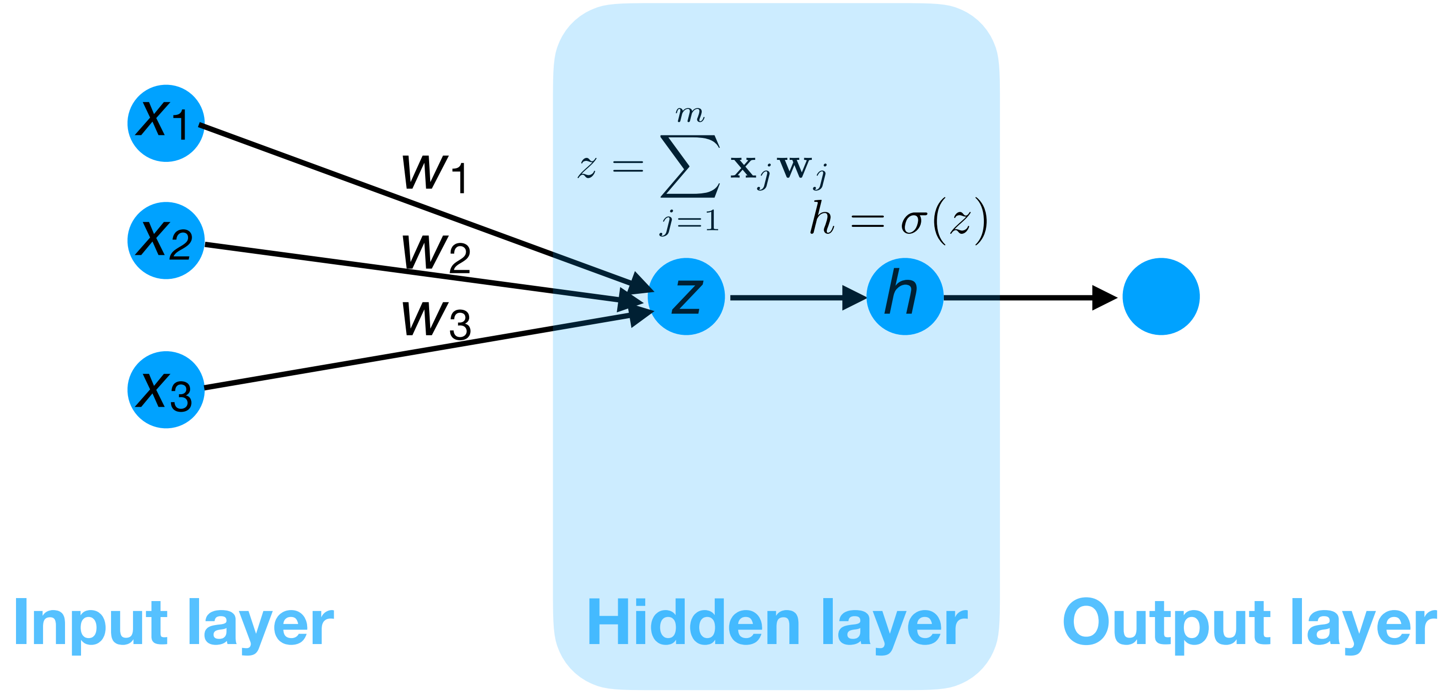
Example: feed-forward NN

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



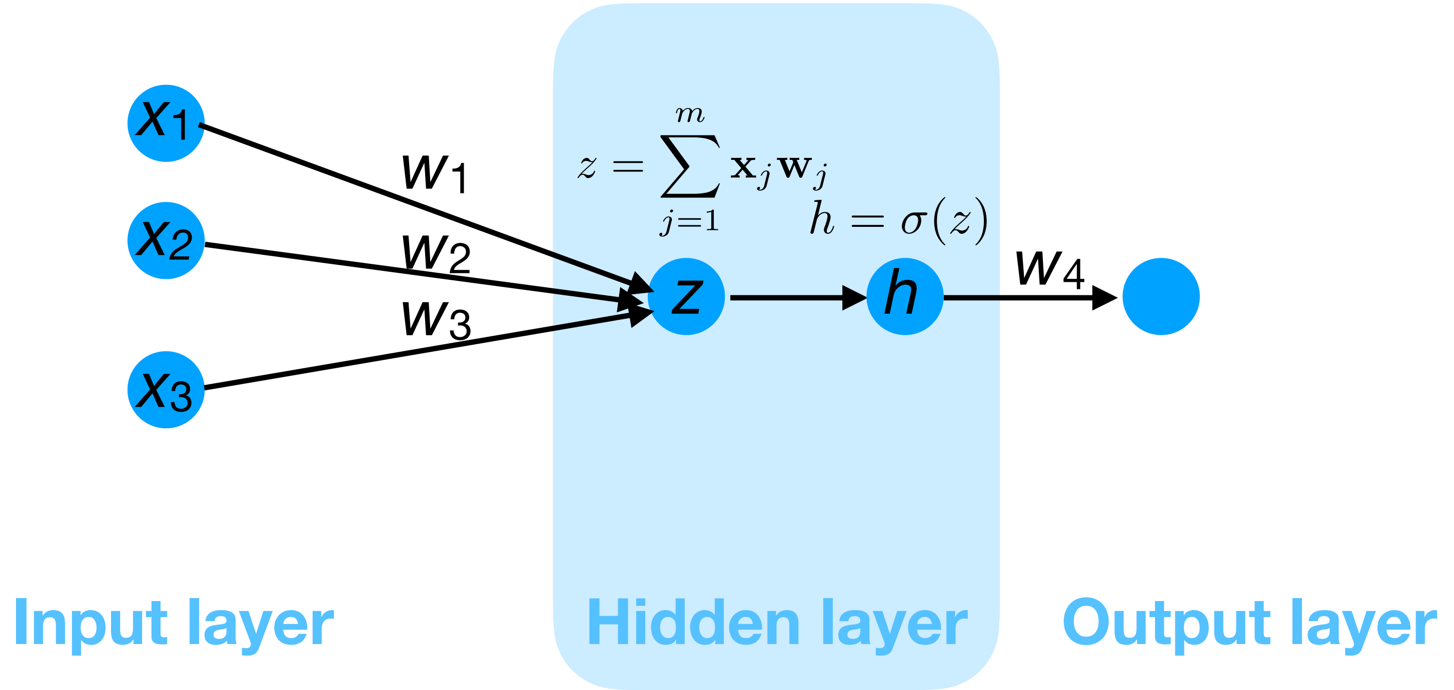
Example: feed-forward NN

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



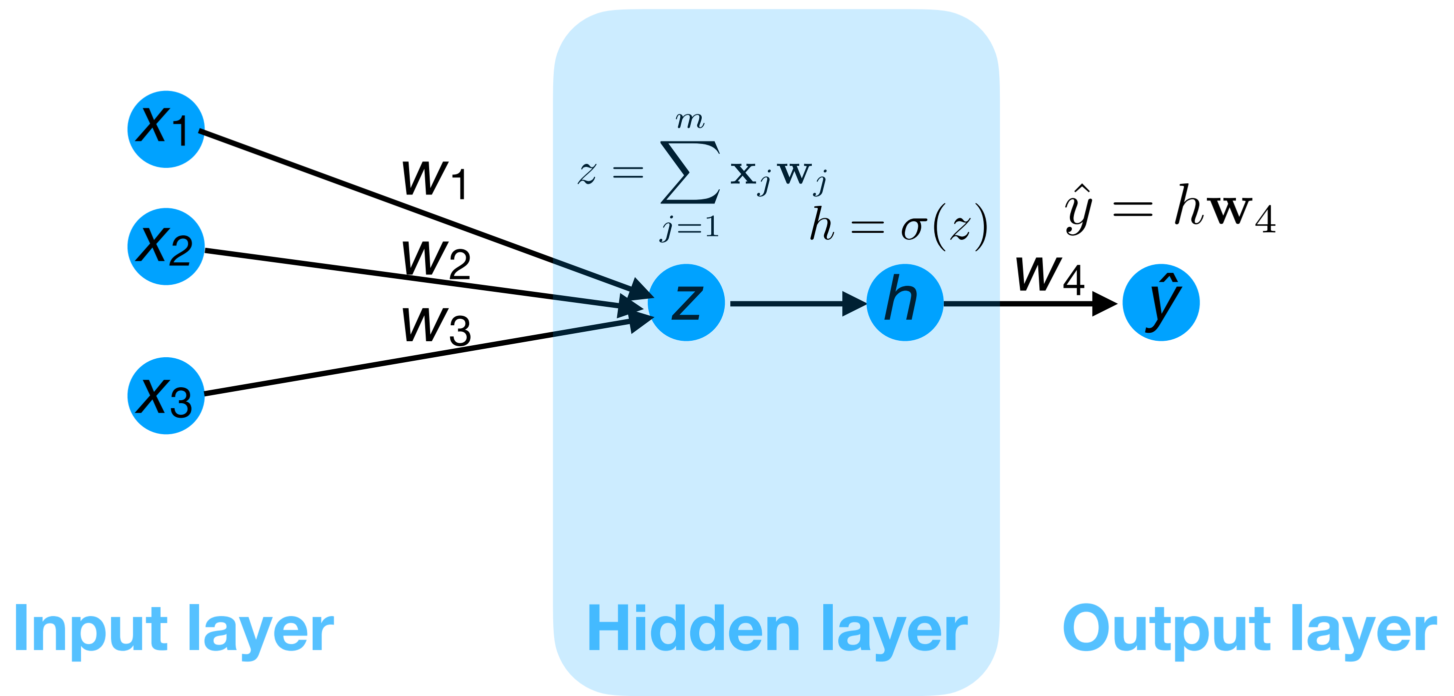
Example: feed-forward NN

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



Example: feed-forward NN

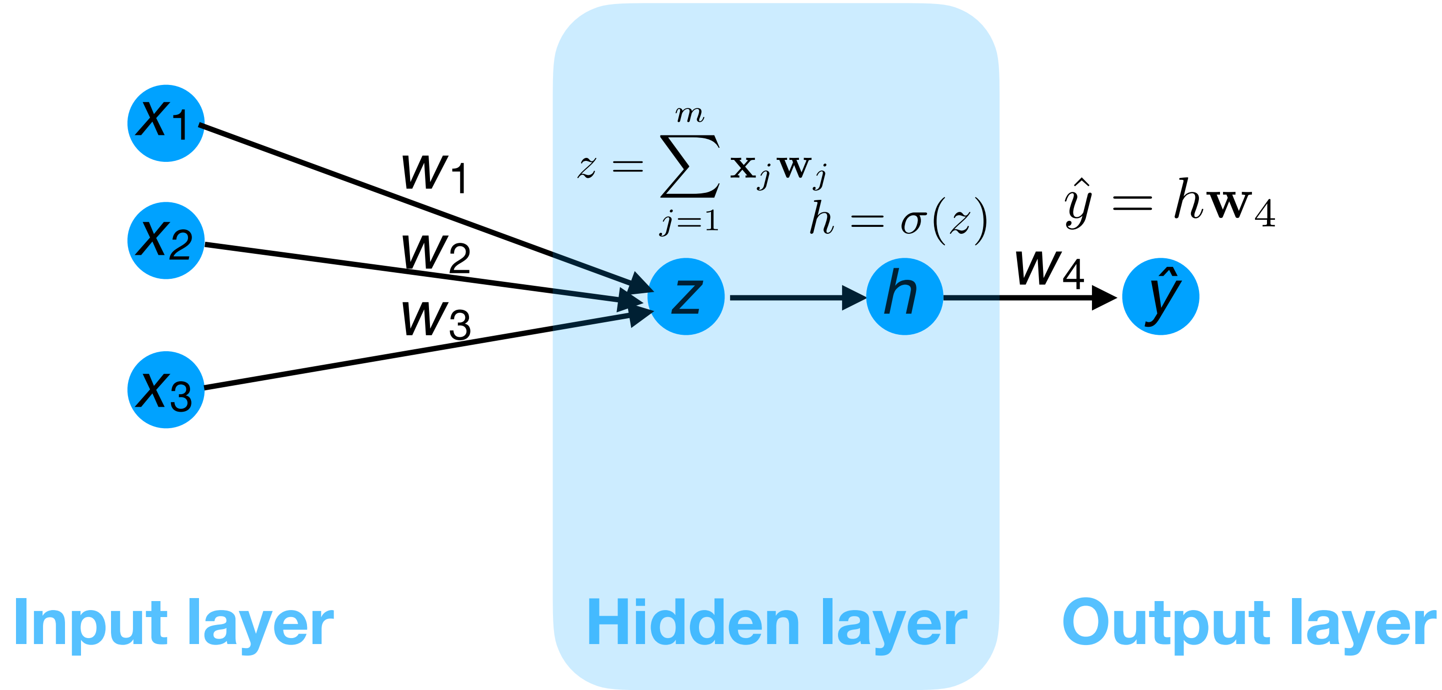
- …to obtain the output $\hat{y}$:



Example: feed-forward NN

- In aggregate, our NN below computes the function:

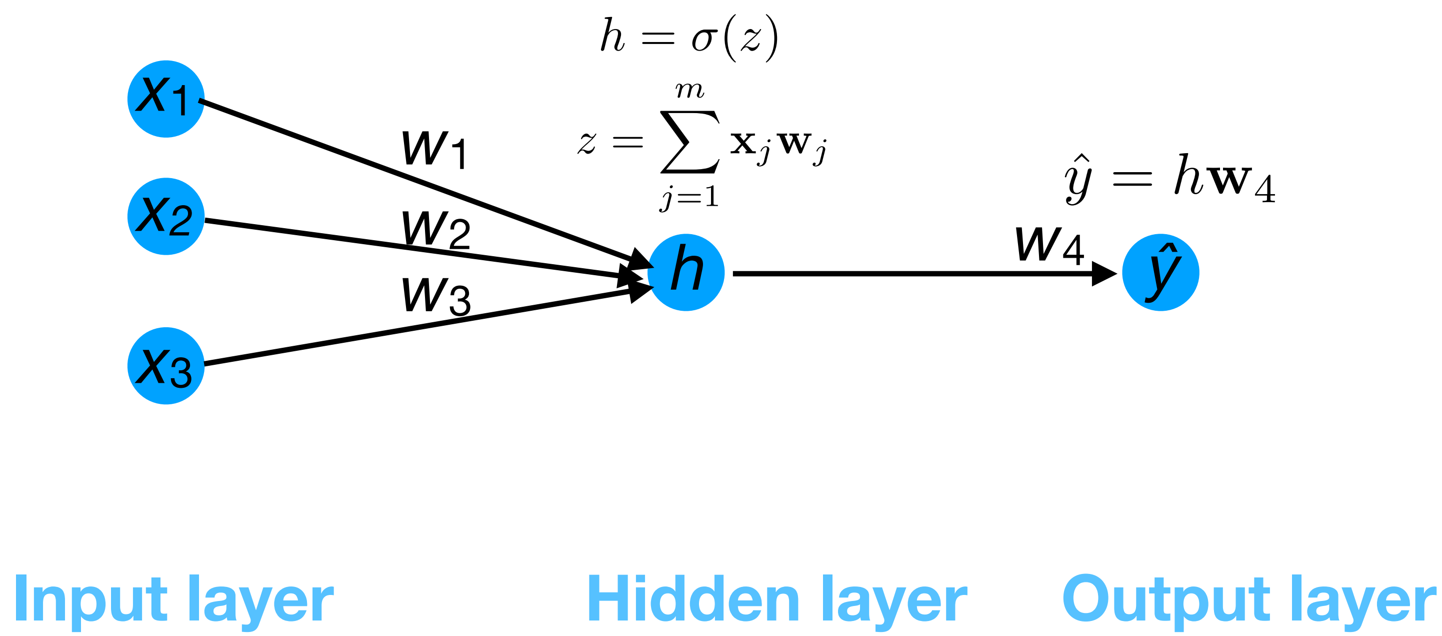
$$\hat{y} = g(\mathbf{x}) = \mathbf{w}_4 \sigma \left(\sum_{j=1}^{3} \mathbf{x}_j \mathbf{w}_j \right)$$



Example: feed-forward NN

- More commonly, we represent this as:

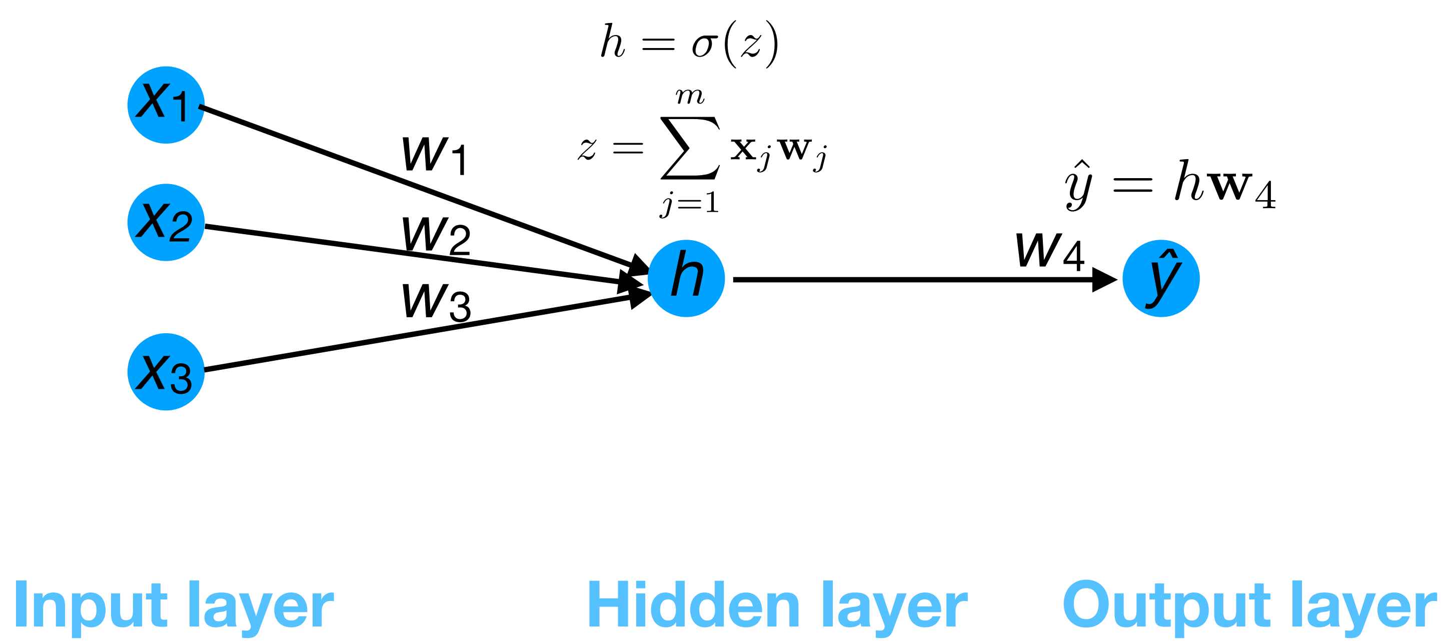
$$\hat{y} = g(\mathbf{x}) = \mathbf{w}_4 \sigma \left(\sum_{j=1}^{3} \mathbf{x}_j \mathbf{w}_j \right)$$



Exercise: feed-forward NN

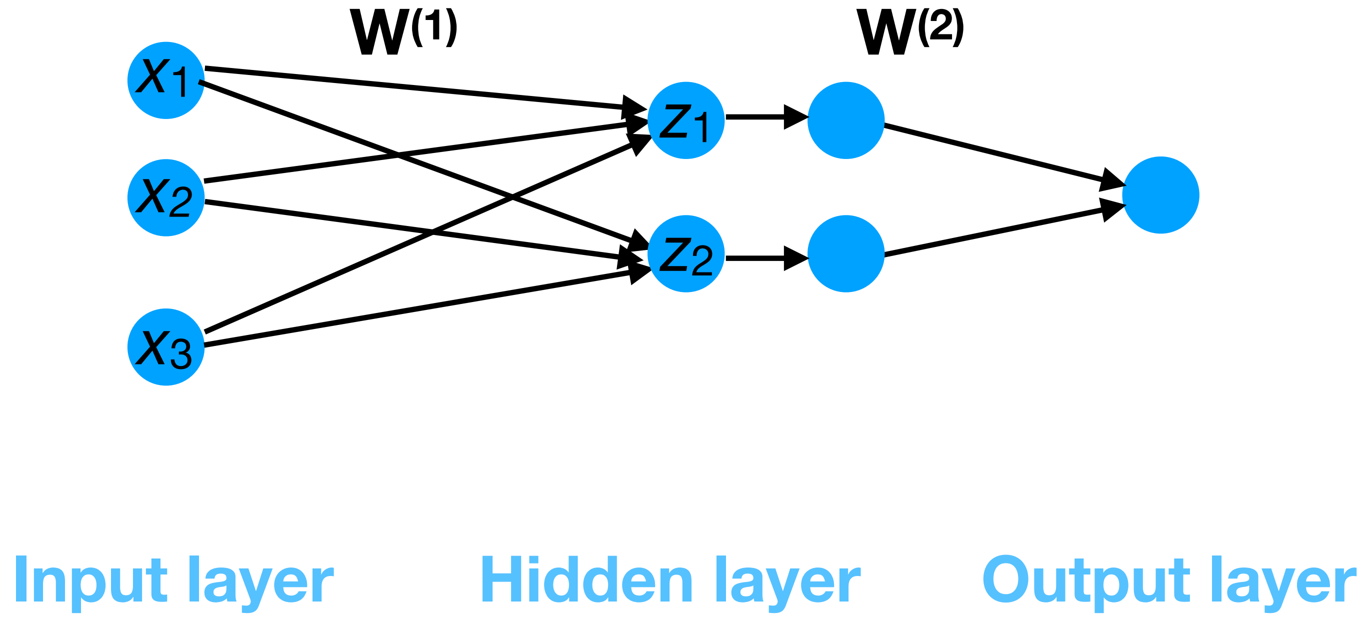
- What will $\hat{y}$ be for $\mathbf{x}$ = $[1\ 0\ -2]^T$, $\mathbf{w}_1$=1, $\mathbf{w}_2$=2, $\mathbf{w}_3$=-1.5, and $\mathbf{w}_4$=-1? Assume ReLU is the activation function.

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



Feed-forward NN

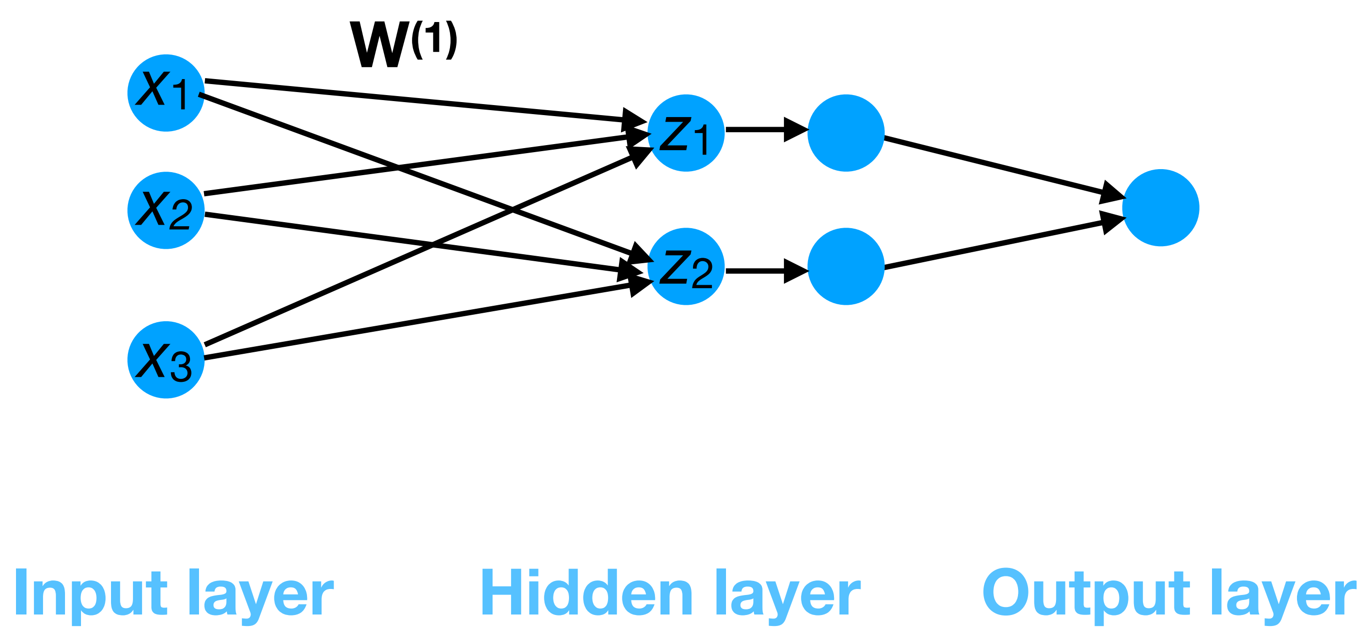
- Neural networks can have multiple neurons per layer.
- Between each adjacent pair of layers (input-hidden and hidden-output), there is a *matrix* of (synaptic) weights:



Feed-forward NN

- We can compute the pre-activation values **z** of the hidden layer as:

$$\mathbf{z} = \mathbf{W}^{(1)}\mathbf{x}$$

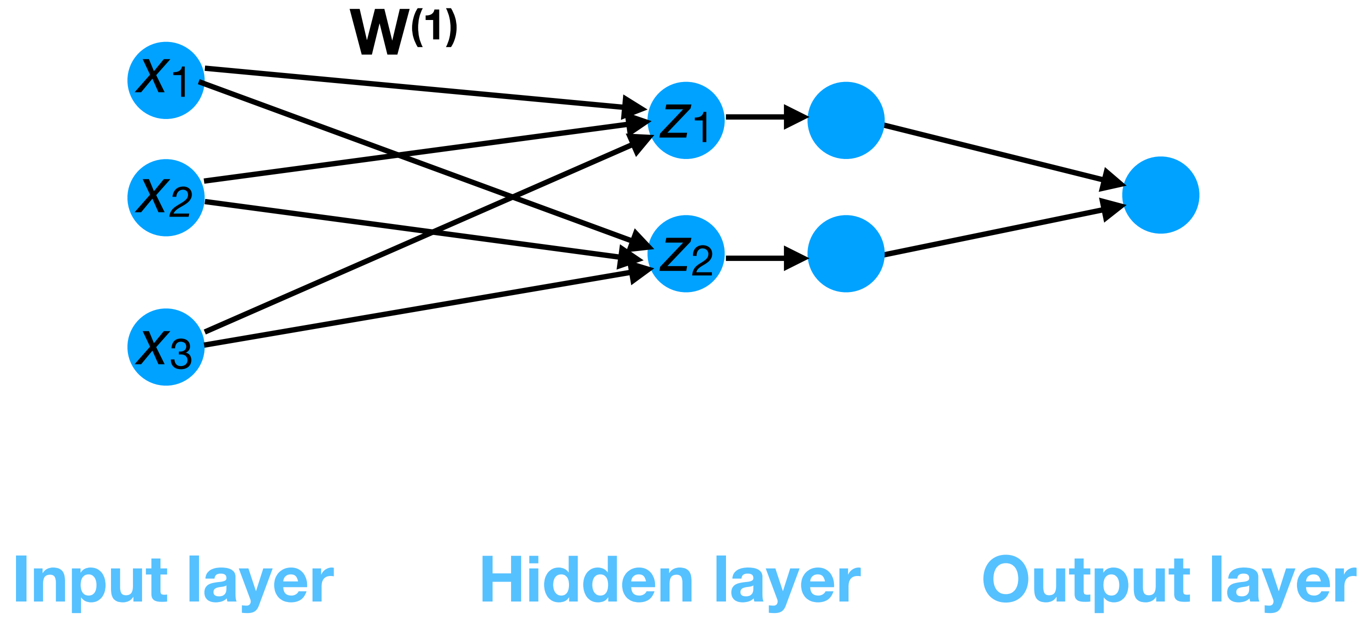


Feed-forward NN

- We can compute the pre-activation values **z** of the hidden layer as:

$$\mathbf{z} = \mathbf{W}^{(1)}\mathbf{x}$$

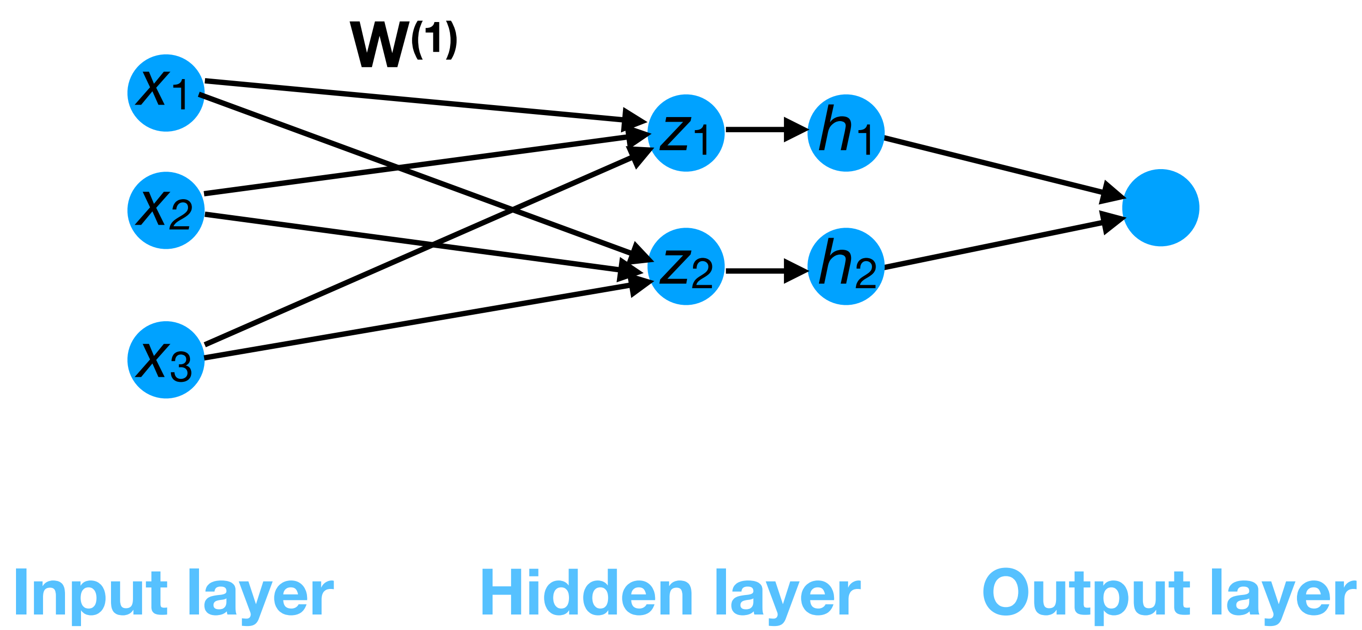
$\mathbf{W}^{(1)}$ is 2 x 3.



Feed-forward NN

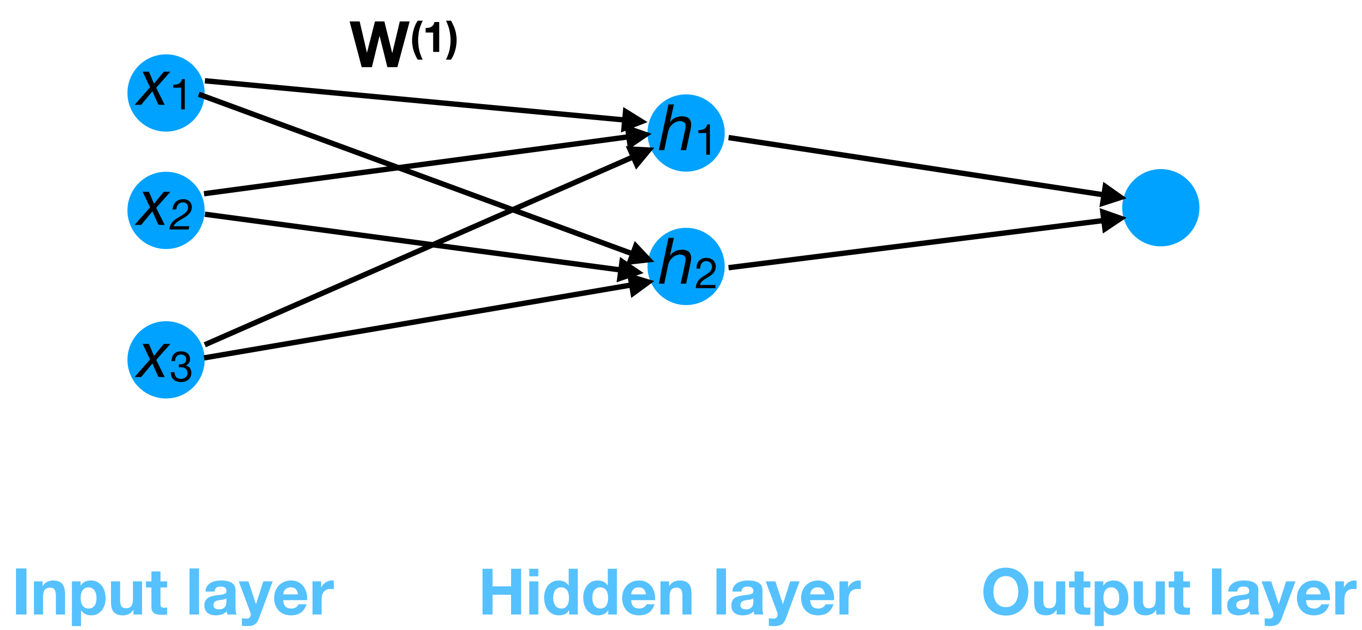
- We can then pass **z** to the activation function σ and compute the hidden neuron values **element-wise**, i.e.:

$$\mathbf{h}_j = \sigma(\mathbf{z}_j)$$



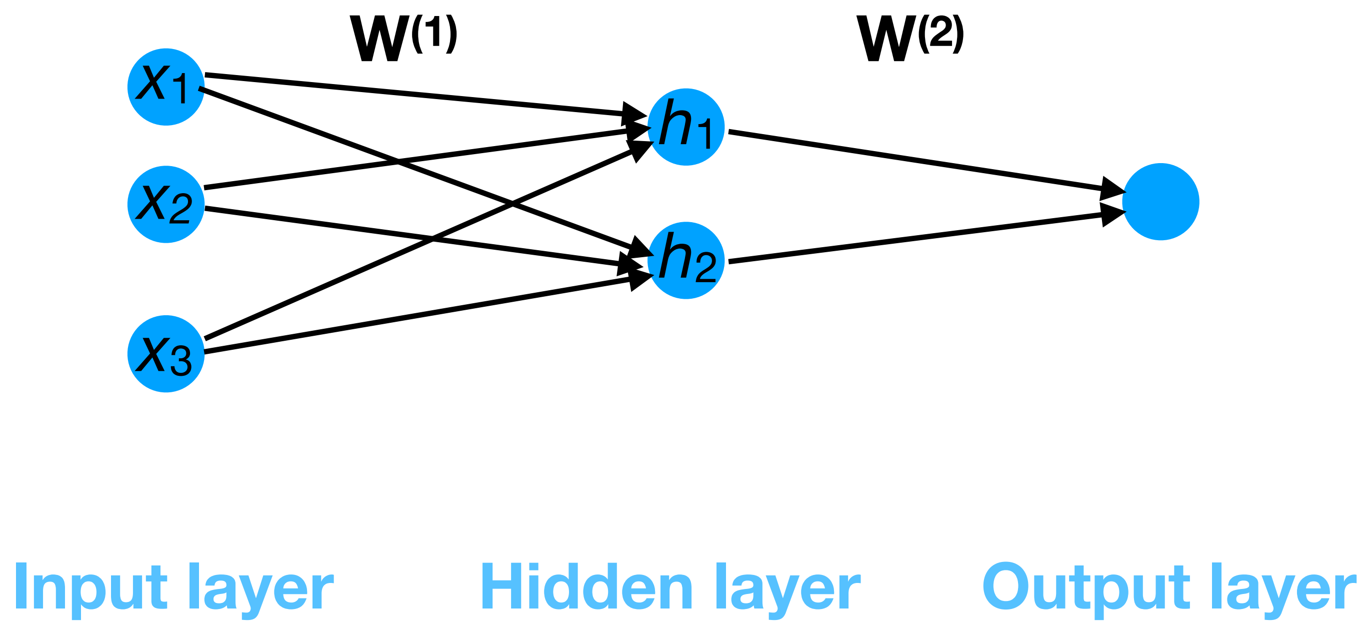
Feed-forward NN

- We typically do not shown the **z** layer explicitly; it is subsumed into the **h** layer to avoid clutter.



Feed-forward NN

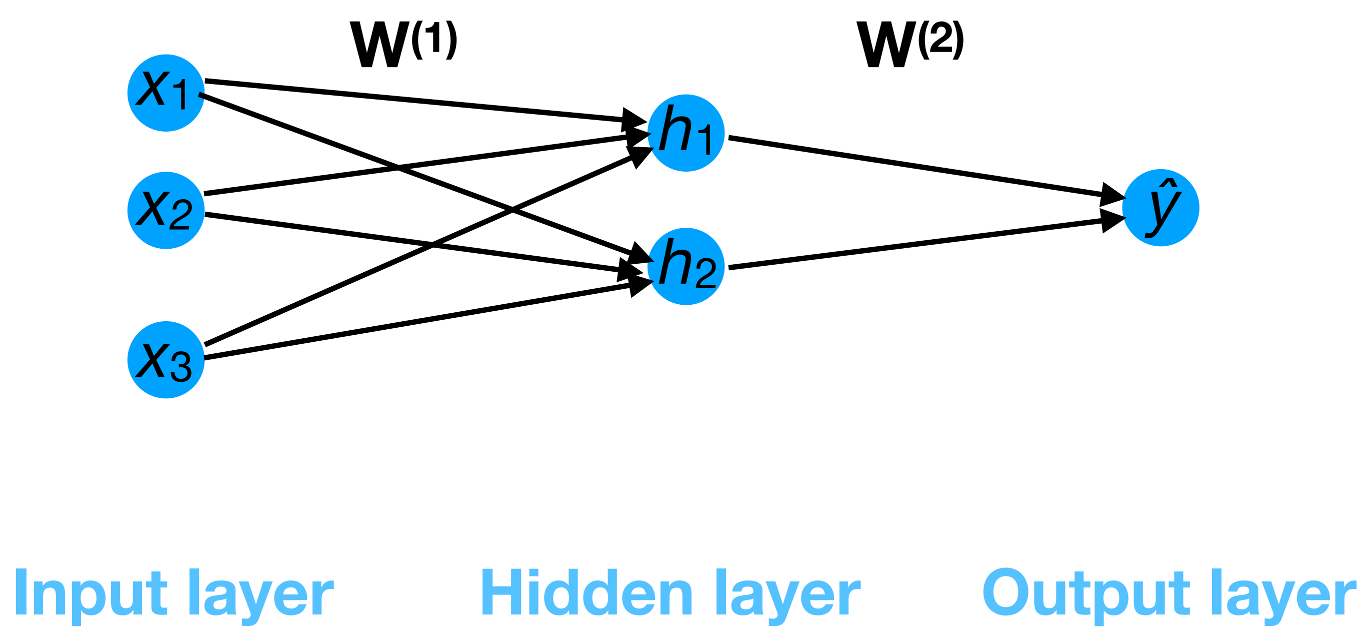
- Next, we pass **h** to the next layer…



Feed-forward NN

- …and compute the product:

$$\hat{\mathbf{y}} = \mathbf{W}^{(2)}\mathbf{h}$$

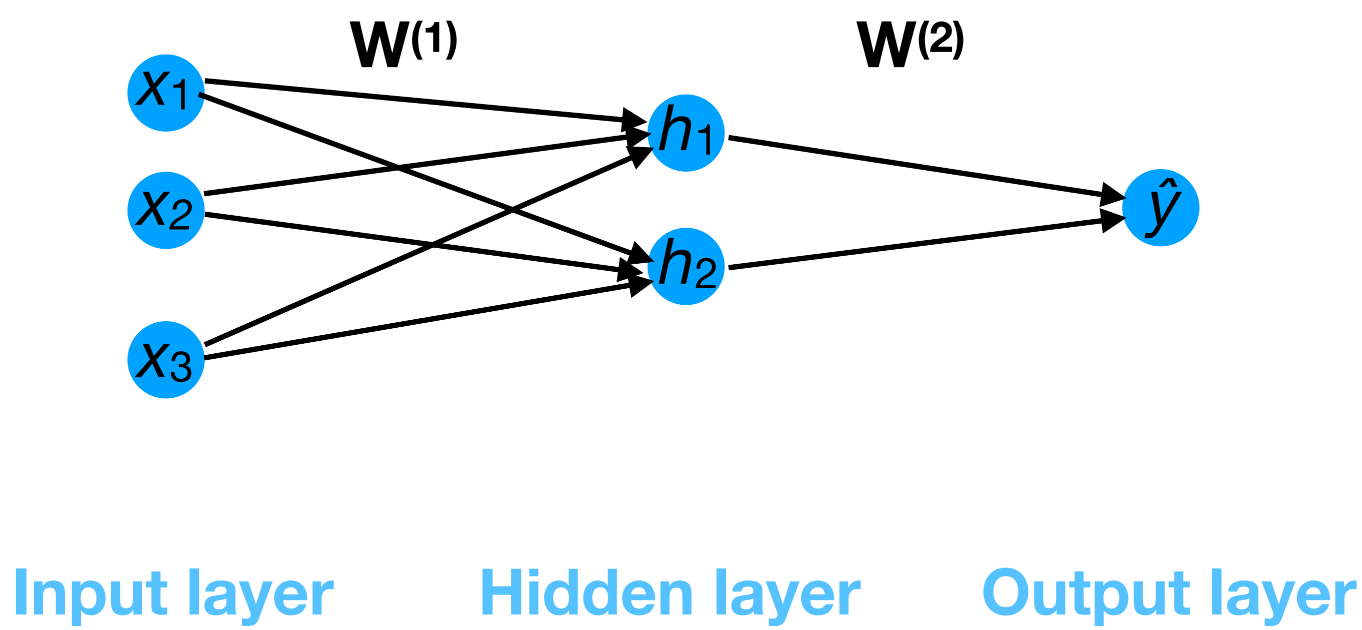


Feed-forward NN

- …and compute the product:

$$\hat{\mathbf{y}} = \mathbf{W}^{(2)}\mathbf{h}$$

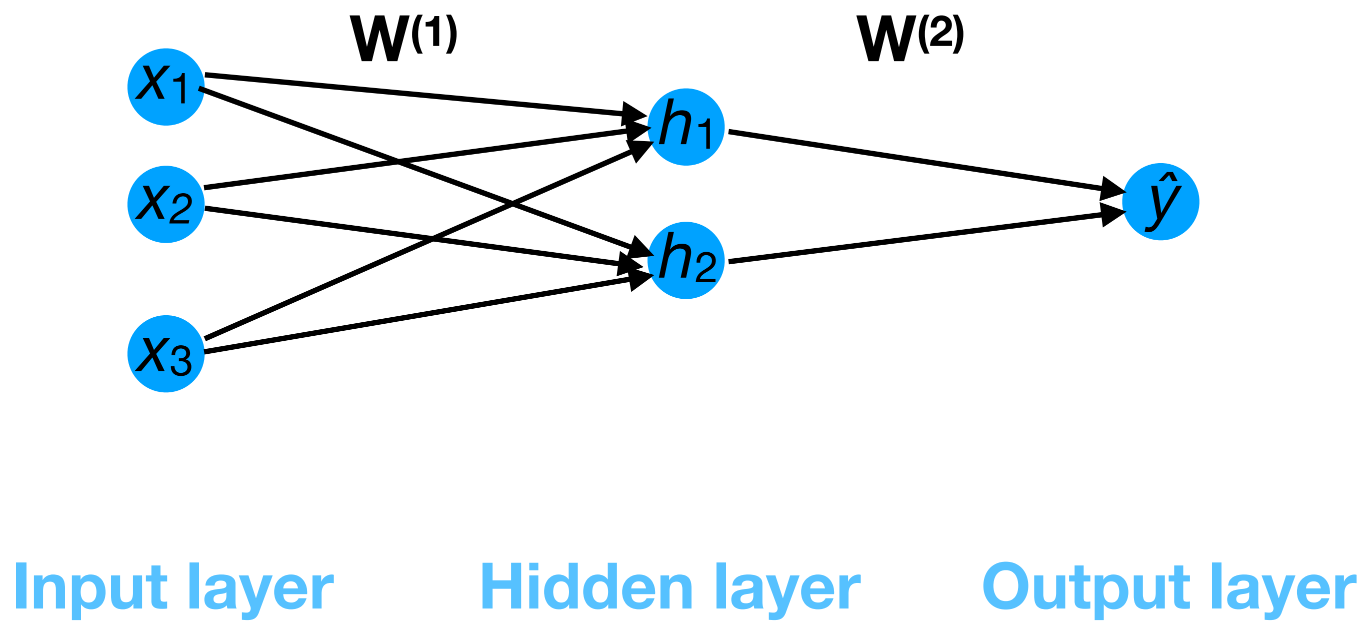
$W^{(2)}$ is 1 x 2.



Feed-forward NN

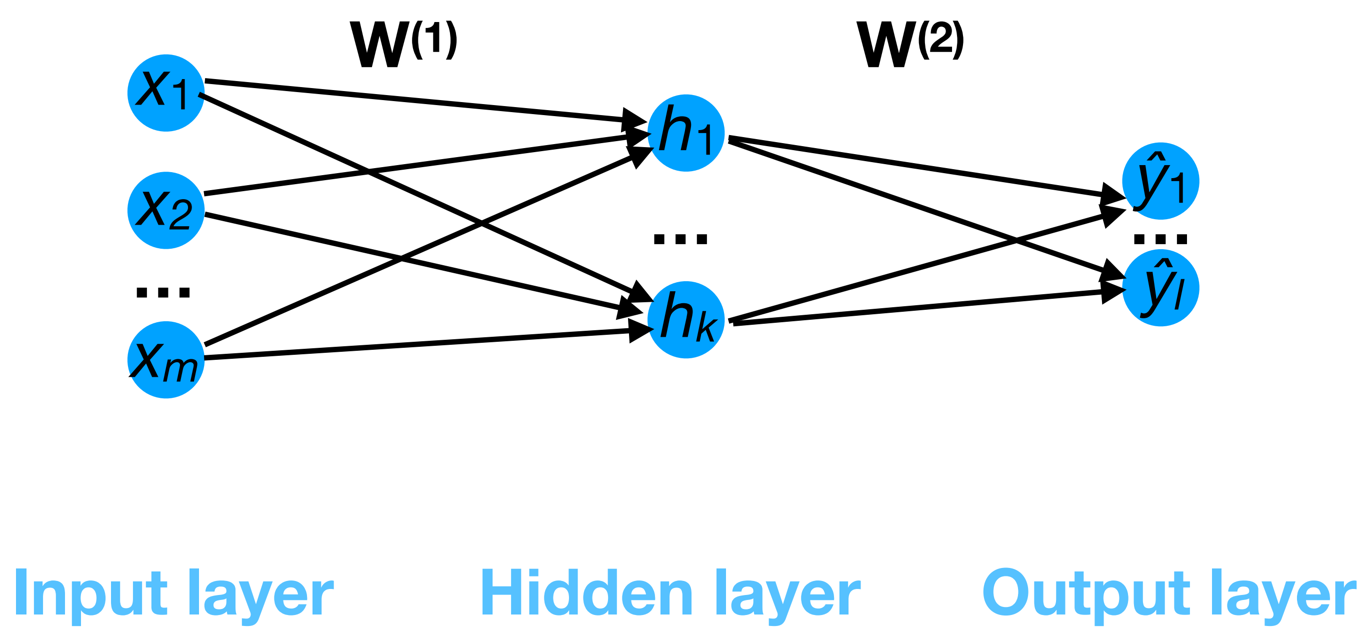
- Note that the final layer could also have an activation function (if we wanted one), e.g.:

$$\hat{\mathbf{y}} = \sigma\left(\mathbf{W}^{(2)}\mathbf{h}\right)$$



Multiple output neurons

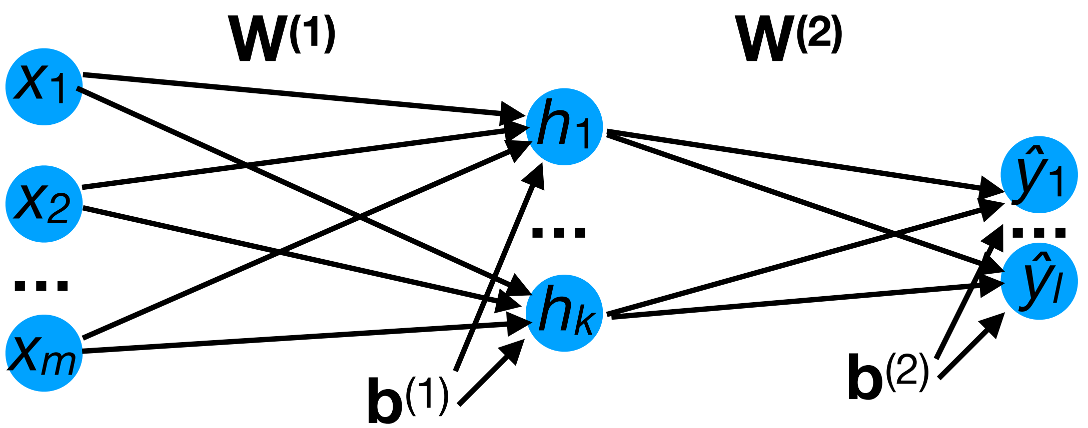
- We can also have a NN with multiple output neurons, e.g.:



Bias terms

- We typically include a **bias term** for every neuron, so that the layers' values are computed as:

$$\mathbf{z} = \mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)} \qquad \textbf{and} \qquad \hat{\mathbf{y}} = \mathbf{W}^{(2)}\mathbf{h} + \mathbf{b}^{(2)}$$



Input layer Hidden layer Output layer

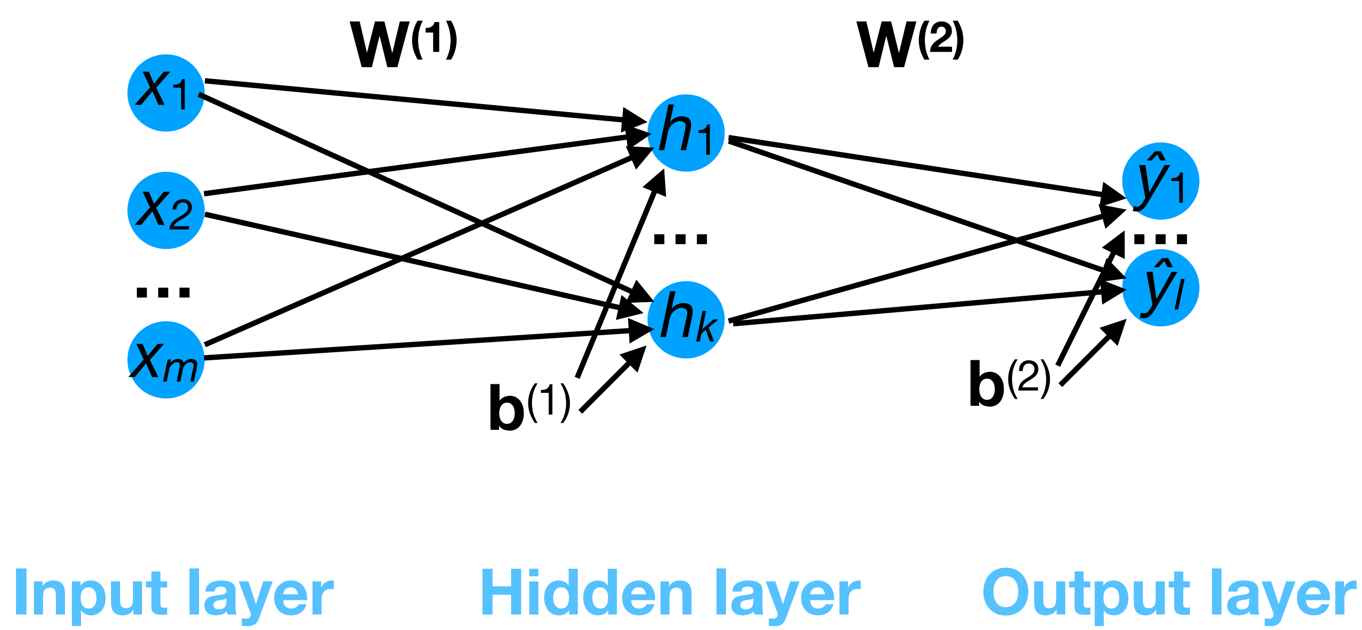
Exercise: bias terms

- What will $\hat{\mathbf{y}}$ be for $\mathbf{x} = [1\ 0\ {-2}]^T$,

$$\mathbf{W}^{(1)} = \begin{bmatrix} 1 & 0 & 1 \\ -1 & 2 & 3 \end{bmatrix} \quad \mathbf{b}^{(1)} = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$$

$$\mathbf{W}^{(2)} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \quad \mathbf{b}^{(2)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

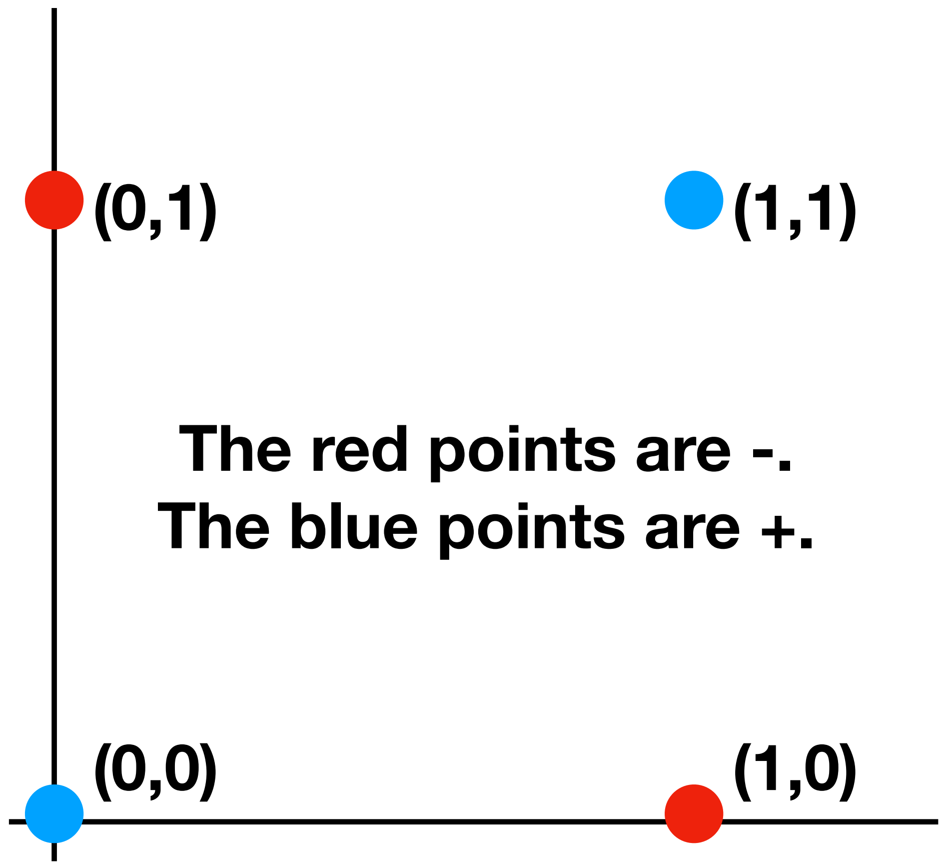
$$\hat{y} = g(\mathbf{x}) = \mathbf{W}^{(2)}\sigma\left(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}\right) + \mathbf{b}^{(2)}$$



Neural networks for non-linear classification

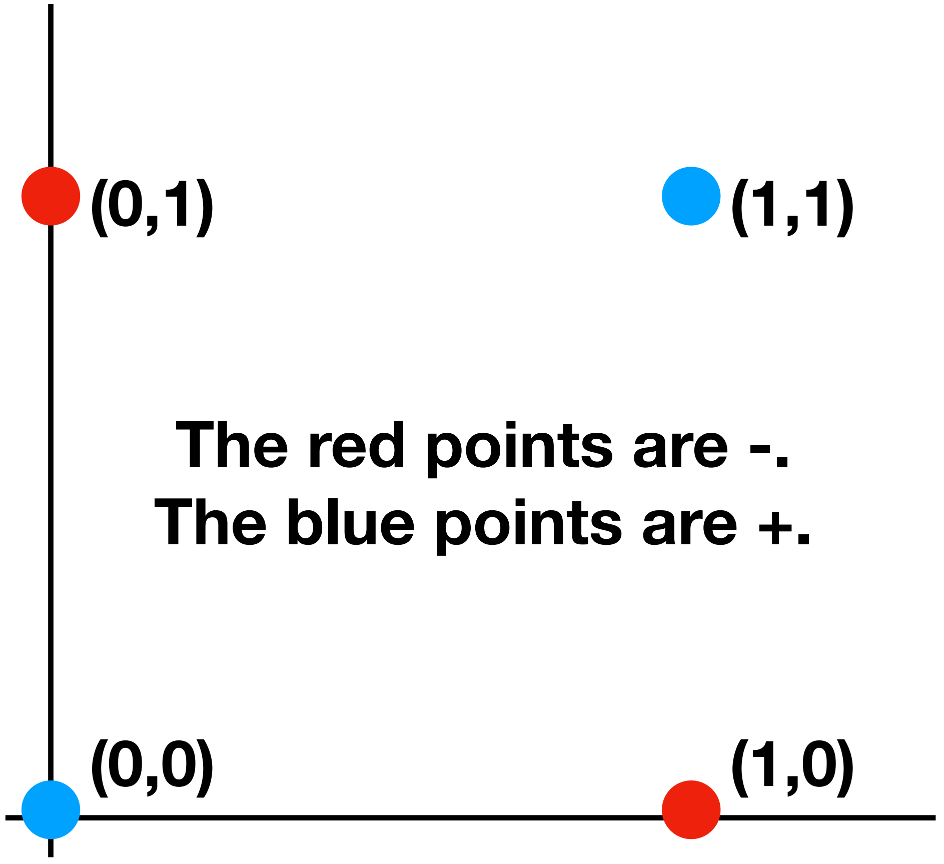
XOR problem

- Recall that no linear decision boundary (e.g., linear SVM, linear regression) can solve the XOR problem.



XOR problem

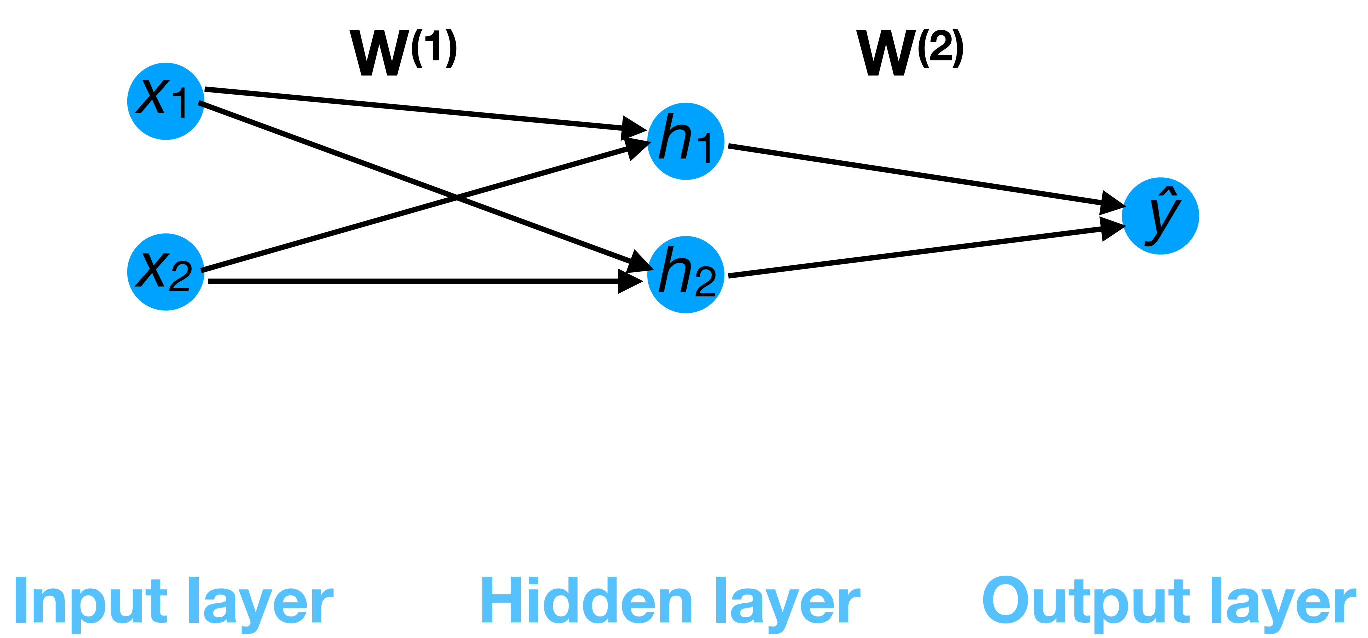
- Recall that no linear decision boundary (e.g., linear SVM, linear regression) can solve the XOR problem.



- Let's see how using a hidden layer can help us solve it…

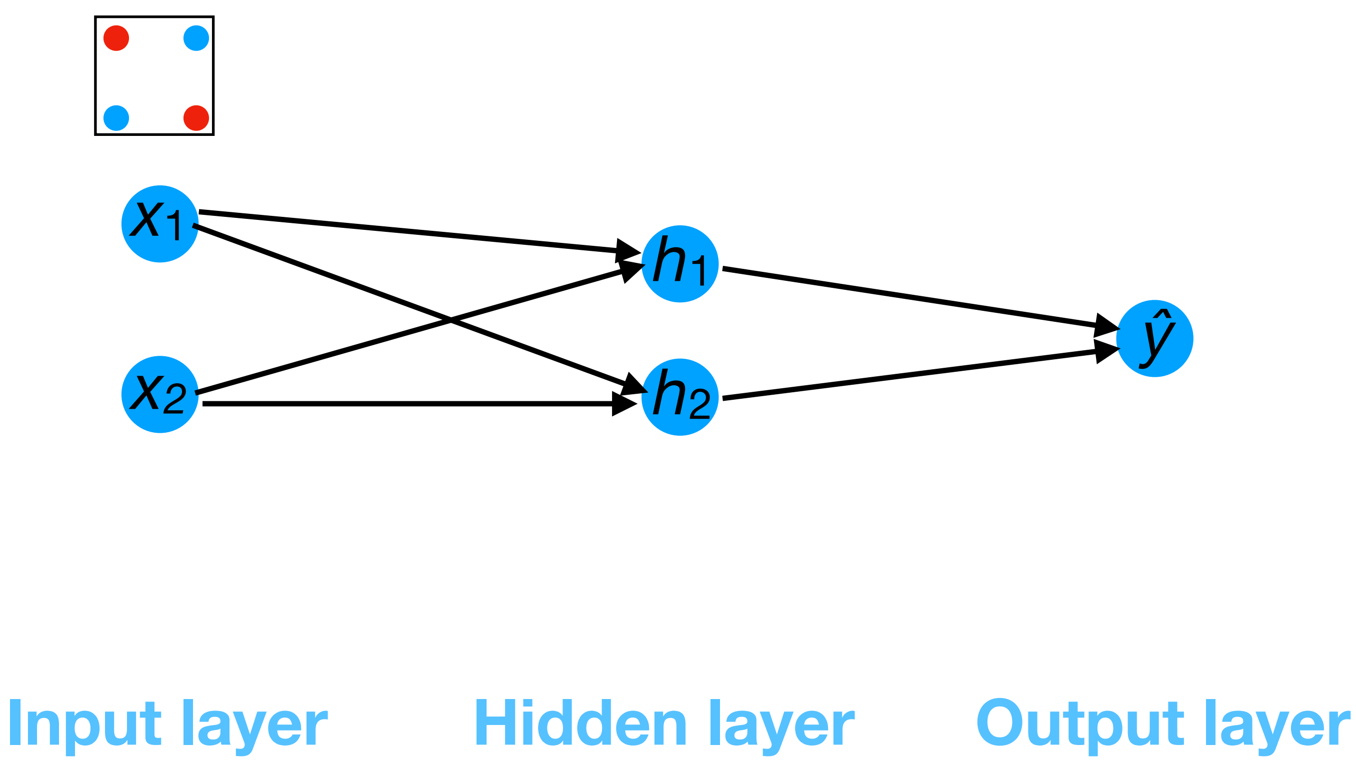
XOR problem

- We want to use a NN to define a function g such that:
  - $g(0,0) = 0$
    $g(0,1) = 1$
    $g(1,0) = 1$
    $g(1,1) = 0$



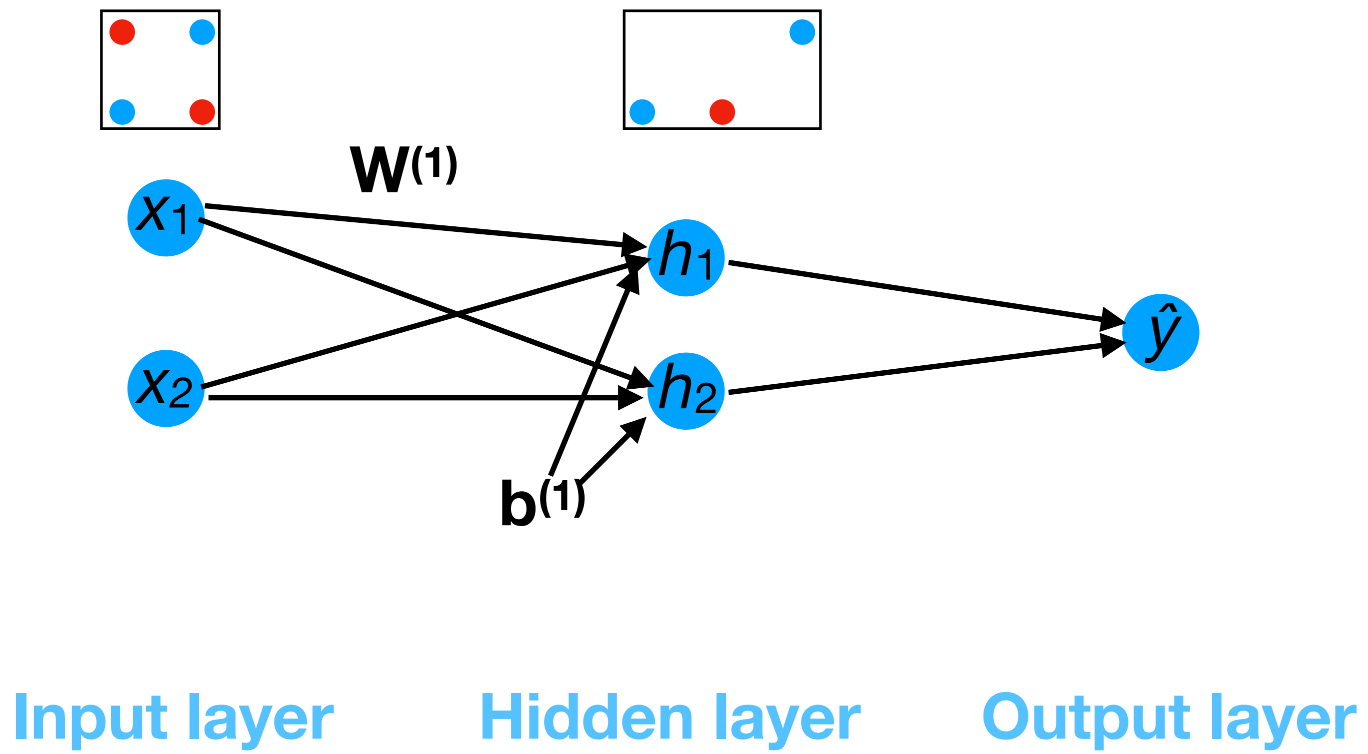
XOR problem

- Here's how a 3-layer NN with a non-linear (ReLU) activation function can solve it:



XOR problem

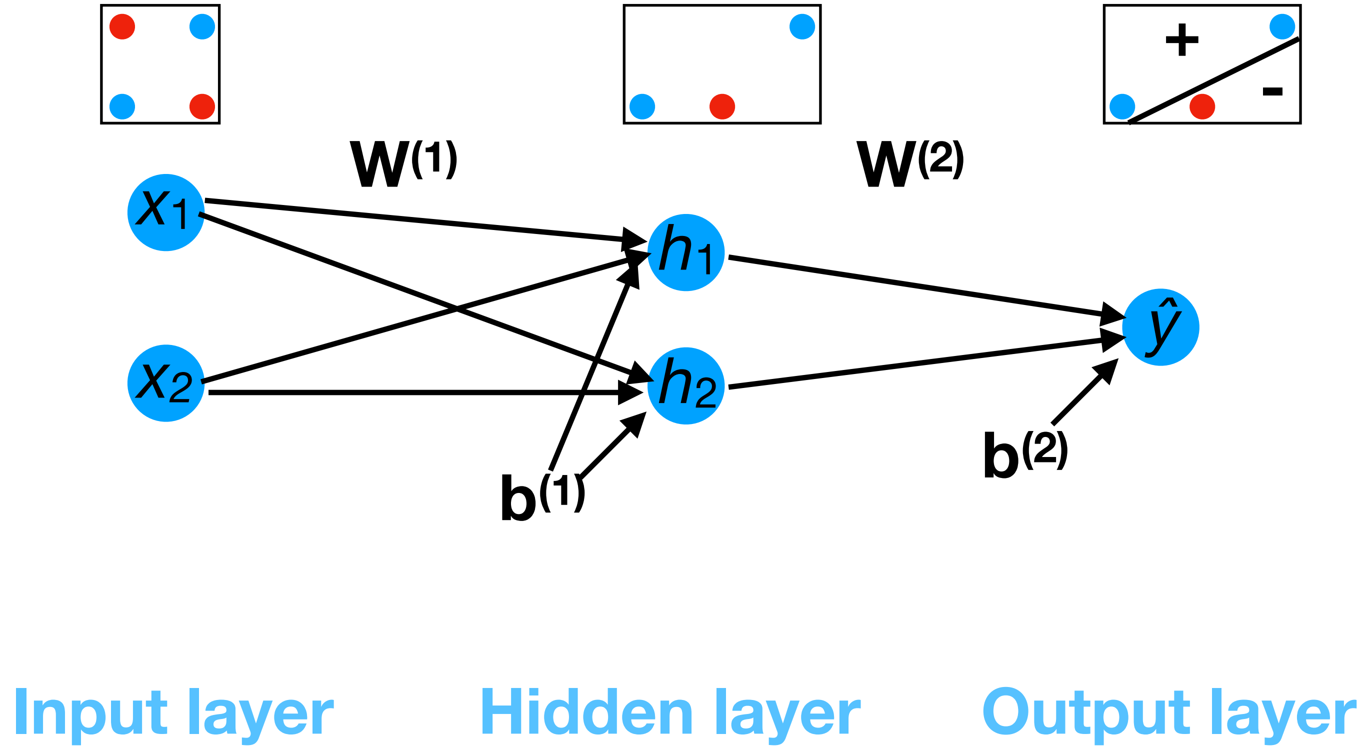
- Here's how a 3-layer NN with a non-linear (ReLU) activation function can solve it:
  - $\mathbf{W}^{(1)}$, $\mathbf{b}^{(1)}$ will "collapse" the two - data points onto one point in the "hidden" 2-D space.



Jacob Whitehill, WPI

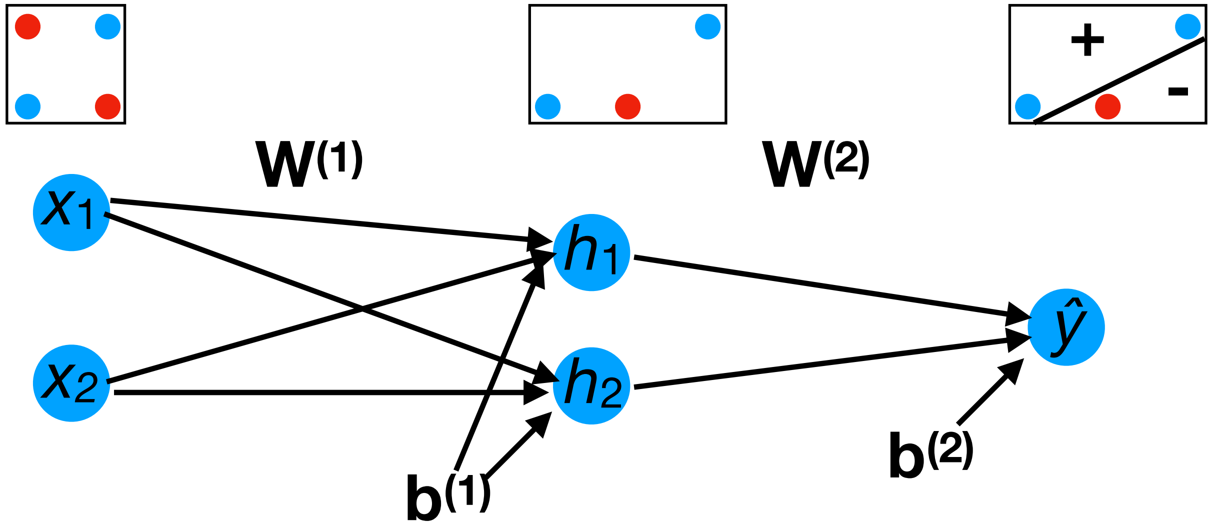
XOR problem

- Here's how a 3-layer NN with a non-linear (ReLU) activation function can solve it:
  - $\mathbf{W}^{(1)}$, $\mathbf{b}^{(1)}$ will "collapse" the two - data points onto one point in the "hidden" 2-D space.
  - Since the + and - data are now linearly separated, $\mathbf{W}^{(2)}$, $\mathbf{b}^{(2)}$ can easily distinguish the two classes.



XOR problem

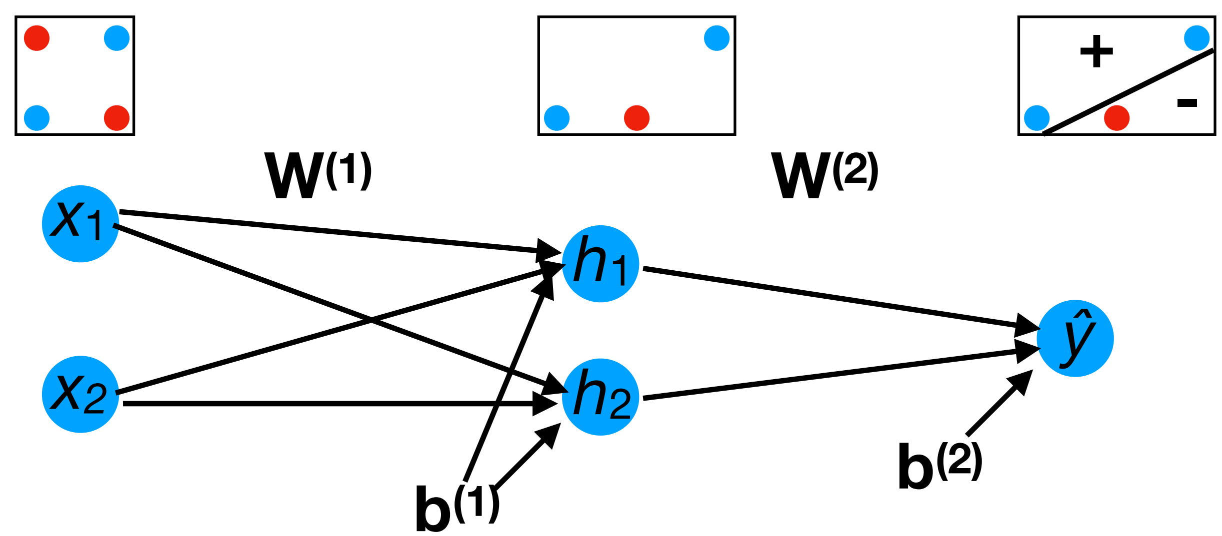
- Here's how a 3-layer NN with a non-linear (ReLU) activation function can solve it:
  - $\mathbf{W}^{(1)}$, $\mathbf{b}^{(1)}$ will "collapse" the two - data points onto one point in the "hidden" 2-D space.
  - Since the + and - data are now linearly separated, $\mathbf{W}^{(2)}$, $\mathbf{b}^{(2)}$ can easily distinguish the two classes.



What values for $W^{(1)}$, $b^{(1)}$ will make the 4 data points linearly separable?
(Hint: set b = [-1 -1]^T; W contains only 1s and 2s.)

XOR problem

- Here's how a 3-layer NN with a non-linear (ReLU) activation function can solve it:
  - $\mathbf{W}^{(1)}$, $\mathbf{b}^{(1)}$ will "collapse" the two - data points onto one point in the "hidden" 2-D space.
  - Since the + and - data are now linearly separated, $\mathbf{W}^{(2)}$, $\mathbf{b}^{(2)}$ can easily distinguish the two classes.



$$\mathbf{W}^{(1)} = \begin{bmatrix} 2 & 2 \\ 1 & 1 \end{bmatrix} \quad \mathbf{b}^{(1)} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

(There are other solutions as well.)

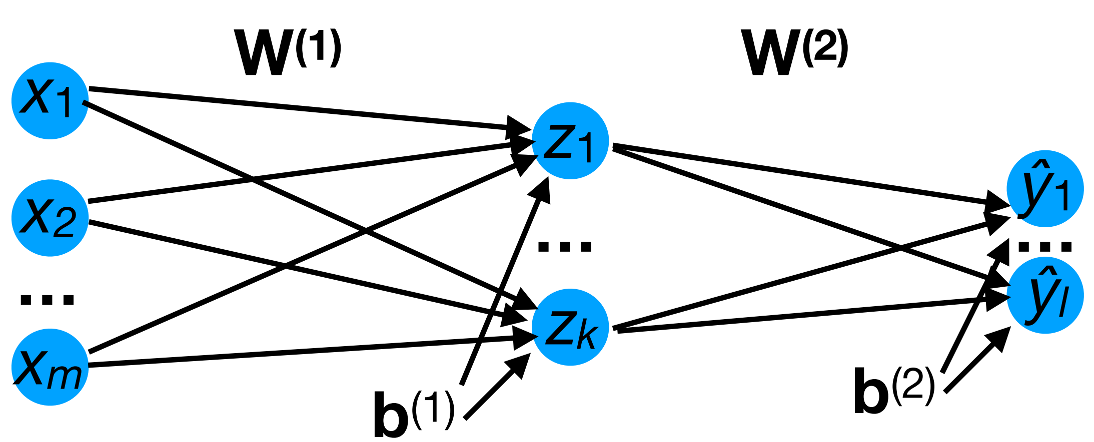
XOR problem

- Note that the ability of the 3-layer NN to solve the XOR problem relies crucially on the non-linear ReLU activation function.
    - Note that other non-linear functions would also work.
- Without non-linearity, a multi-layer NN is no more powerful than a 2-layer network!

Crucial role of non-linearity

- Suppose we define a 3-layer NN **without** non-linearity:

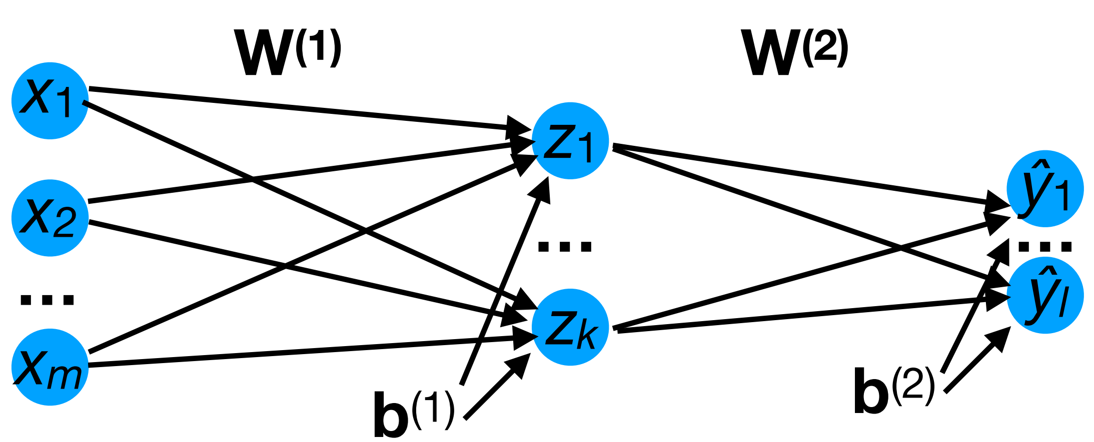
$$g(\mathbf{x}) \quad = \quad \mathbf{W}^{(2)}\left(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}\right) + \mathbf{b}^{(2)}$$



Crucial role of non-linearity

- Then we can simplify g to be:

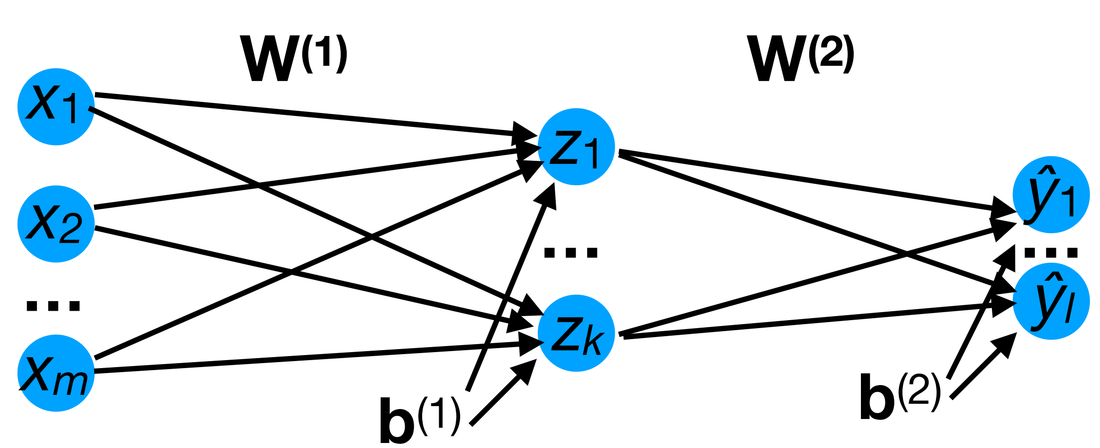
$$g(\mathbf{x}) \quad = \quad \mathbf{W}^{(2)}\left(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}\right) + \mathbf{b}^{(2)}$$



Crucial role of non-linearity

- Then we can simplify *g* to be:

$$
\begin{aligned}
g(\mathbf{x}) &= \mathbf{W}^{(2)}\left(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}\right) + \mathbf{b}^{(2)} \\
&= \left(\mathbf{W}^{(2)}\mathbf{W}^{(1)}\right)\mathbf{x} + \mathbf{W}^{(2)}\mathbf{b}^{(1)} + \mathbf{b}^{(2)}
\end{aligned}
$$



Crucial role of non-linearity

- Then we can simplify g to be:

$$\begin{aligned} g(\mathbf{x}) &= \mathbf{W}^{(2)}\left(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}\right) + \mathbf{b}^{(2)} \\ &= \left(\mathbf{W}^{(2)}\mathbf{W}^{(1)}\right)\mathbf{x} + \mathbf{W}^{(2)}\mathbf{b}^{(1)} + \mathbf{b}^{(2)} \\ &= \mathbf{W}\mathbf{x} + \mathbf{b} \end{aligned}$$