

Neural Networks I

Lecture 18

What's the hype around neural networks?

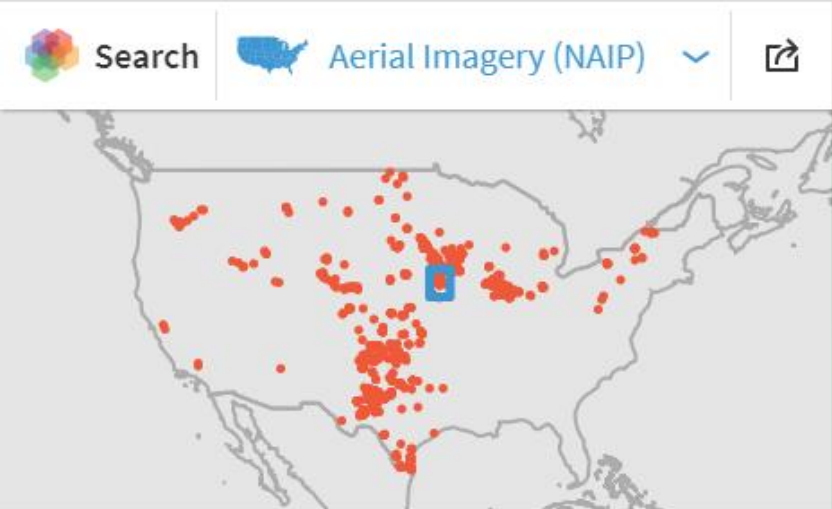
Character/handwriting recognition

Image compression (autoencoders)

Stock market prediction

Credit approval

And some other interesting deep learning applications...



Top 1000 Closest Matches: [Clear Results](#)



Geovisual Search

<https://search.descarteslabs.com/>

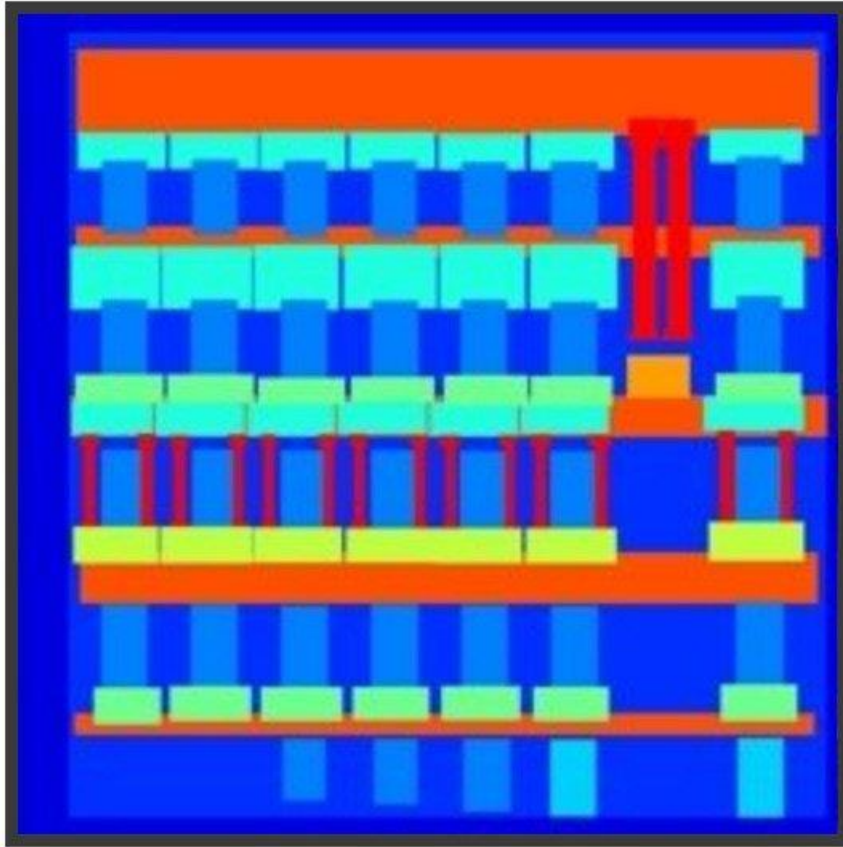


Image-to-image translation

TOOL

- background
- wall
- door
- window**
- window sill
- window head
- shutter
- balcony
- trim
- cornice
- column
- entrance

INPUT



pix2pix
process

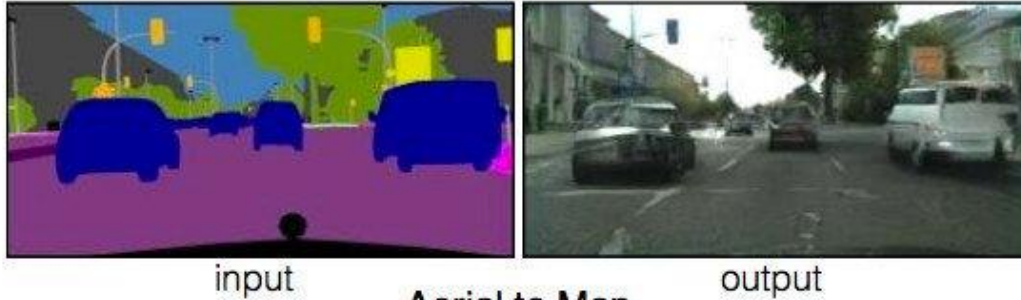
OUTPUT



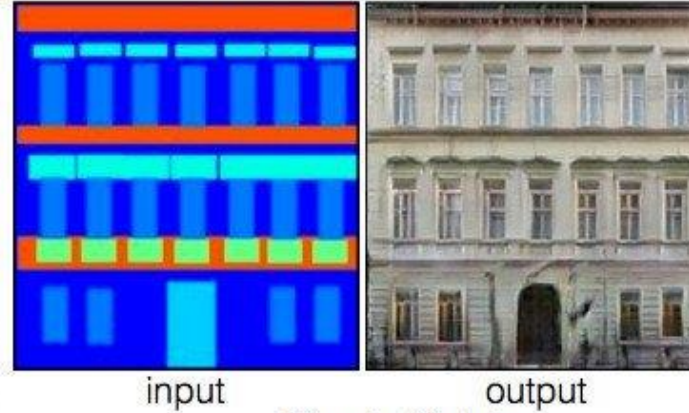
Isola, Phillip, et al. "Image-to-image translation with conditional adversarial networks." arXiv preprint (2017).

Image-to-image translation

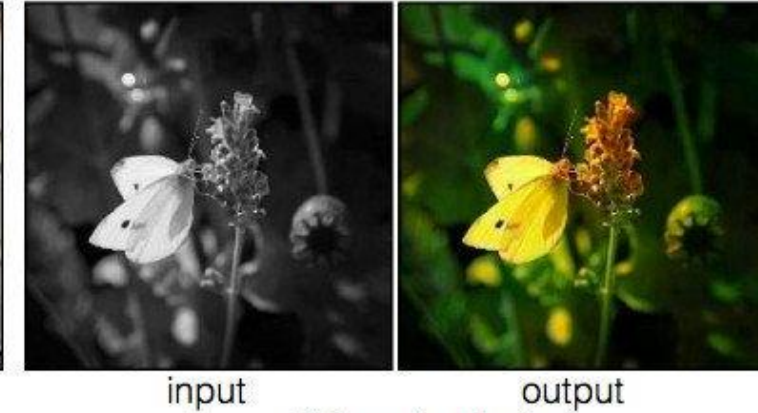
Labels to Street Scene



Labels to Facade



BW to Color



Aerial to Map



Day to Night



Edges to Photo



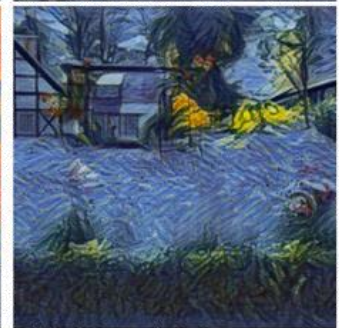
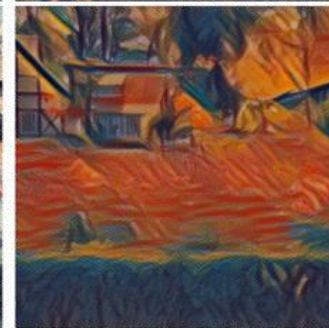
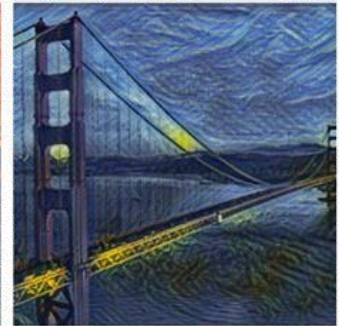
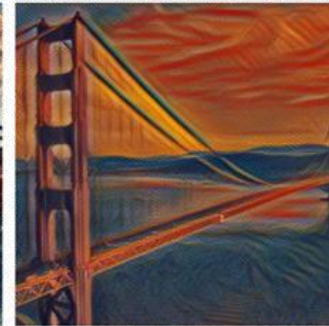
Isola, Phillip, et al. "Image-to-image translation with conditional adversarial networks." arXiv preprint (2017).

Image-to-image translation



Isola, Phillip, et al. "Image-to-image translation with conditional adversarial networks." arXiv preprint (2017).

Image Style Transfer



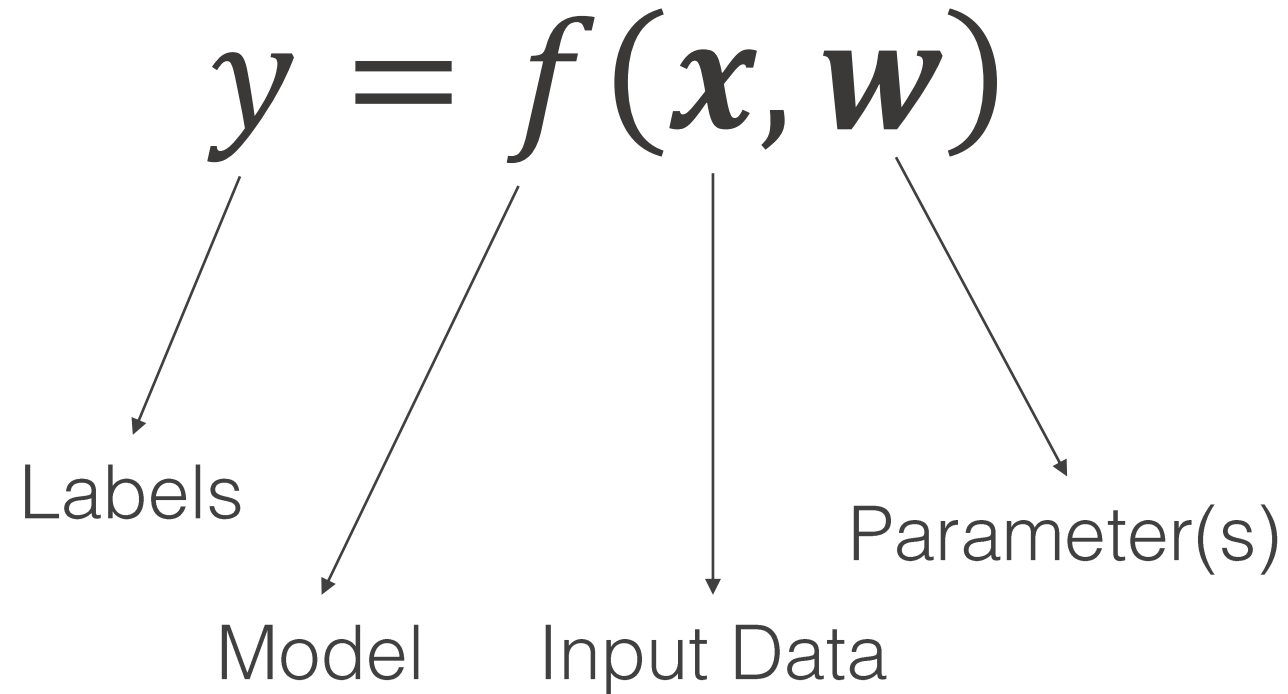
Dumoulin, Vincent, Jonathon Shlens, and Manjunath Kudlur. "A learned representation for artistic style." CoRR, abs/1610.07629 2.4 (2016): 5.

What is a neural network and **how does it work?**

How do we **choose model weights?**
(i.e. how do we fit our model to data)

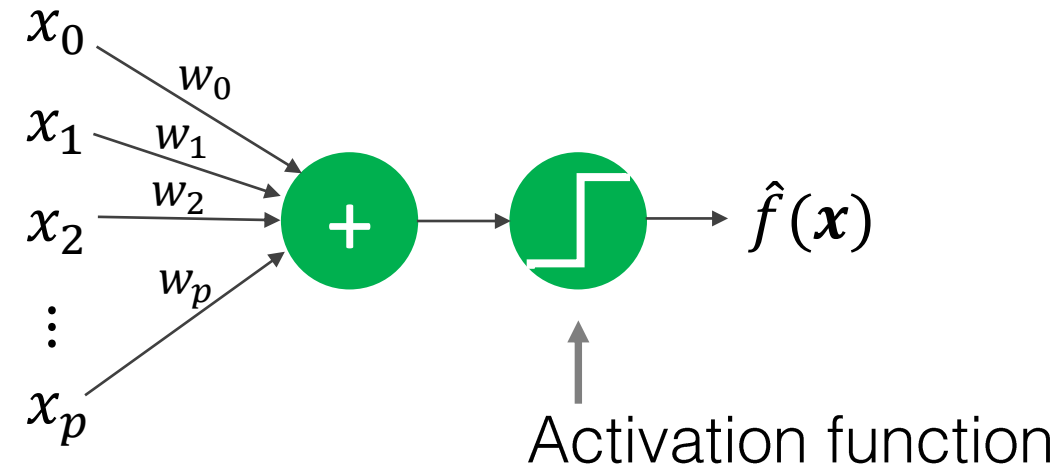
What are the challenges of using neural networks?

Recall our goal in supervised learning



Perceptron

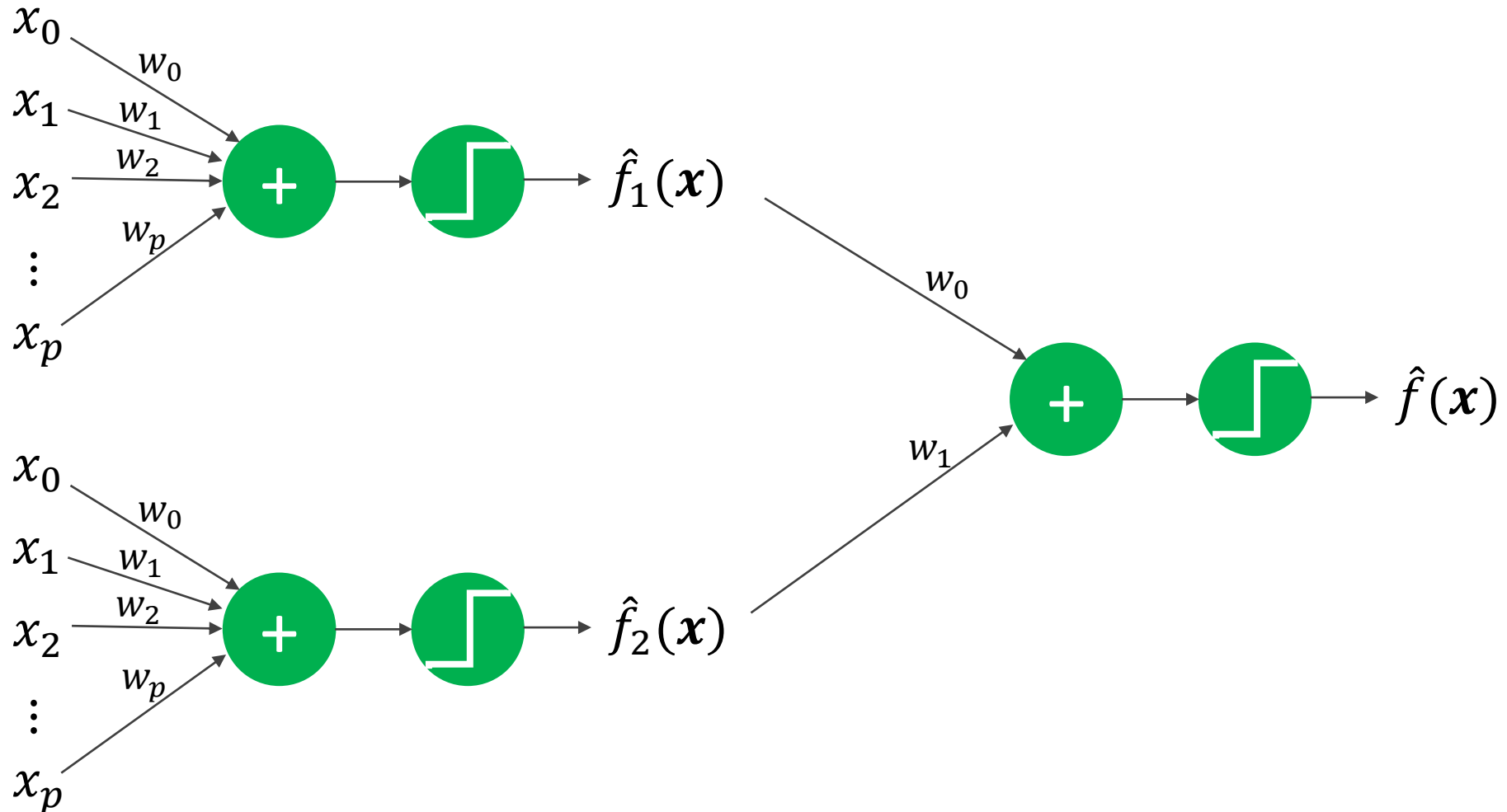
$$\hat{f}(x) = \text{sign} \left(\sum_{i=0}^p w_i x_i \right)$$



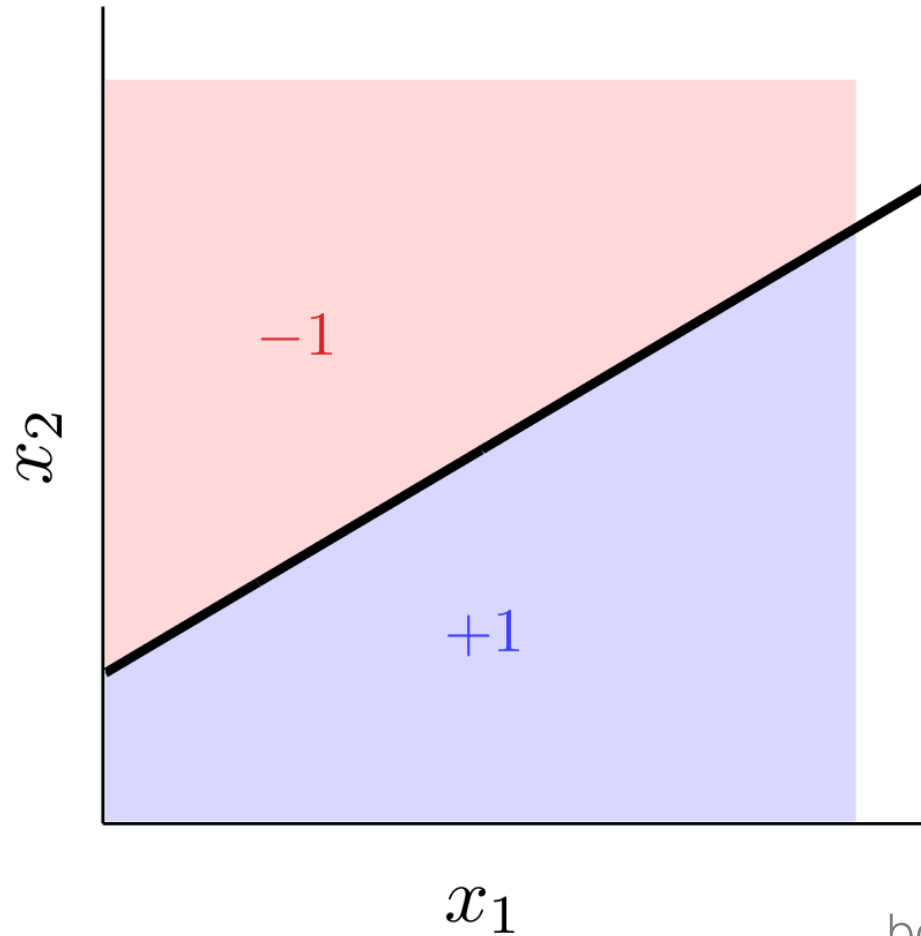
Source: Abu-Mostafa, Learning from Data, Caltech

Multilayer Perceptron

What if we stuck multiple perceptrons together?



Perceptron #1

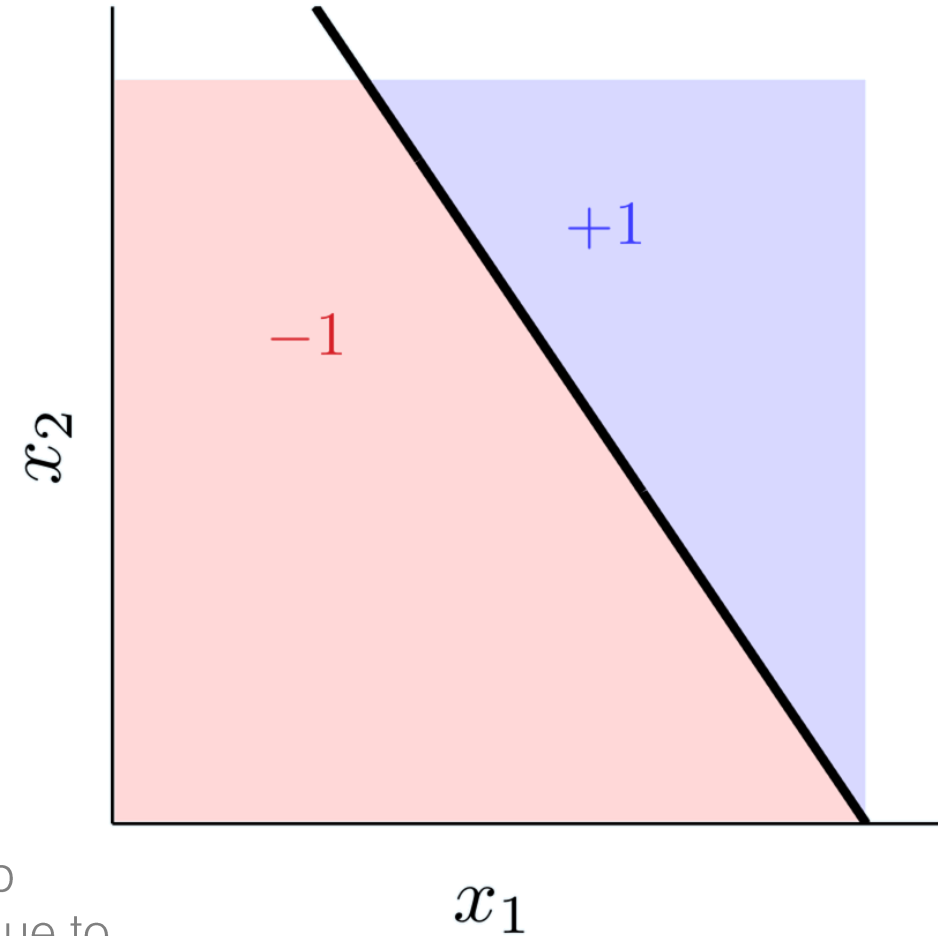


$$\hat{f}_1(\mathbf{x}) = \text{sign}(\mathbf{w}_1^T \mathbf{x})$$

The sharp
boundary is due to
our sign function



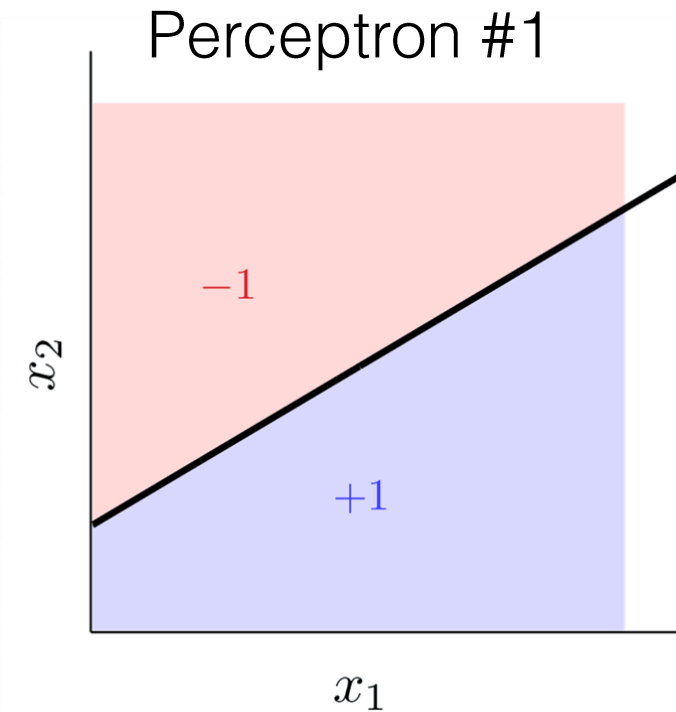
Perceptron #2



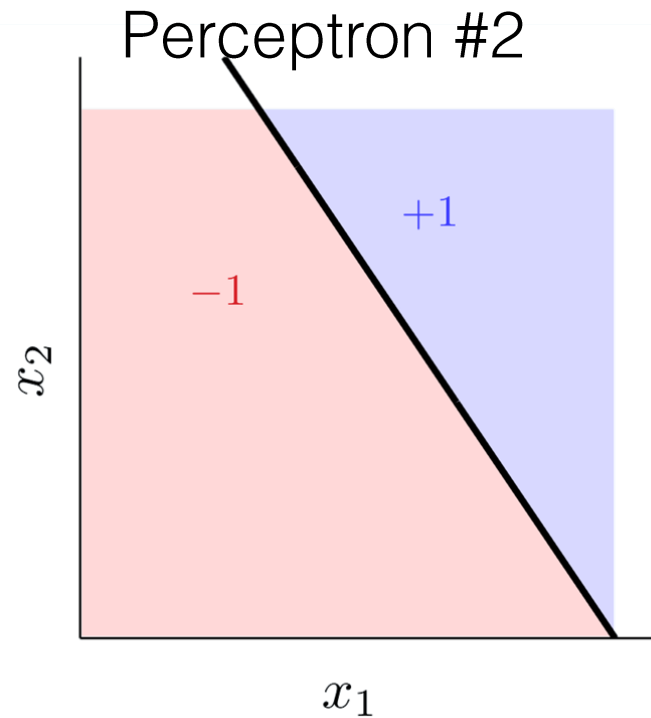
$$\hat{f}_2(\mathbf{x}) = \text{sign}(\mathbf{w}_2^T \mathbf{x})$$

Source: Abu-Mostafa, Learning from Data, Caltech

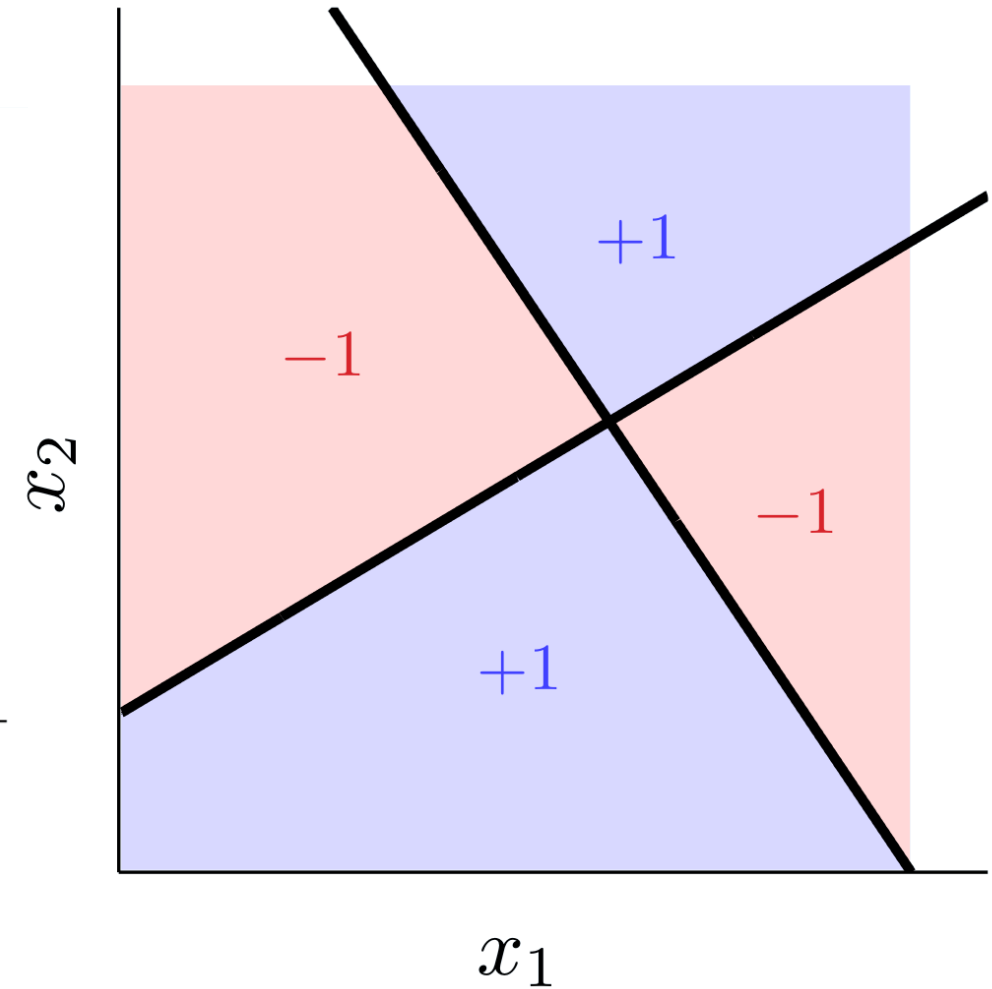
Multilayer perceptron: $\hat{f}(\mathbf{x}) = \begin{cases} +1 & \hat{f}_1(\mathbf{x}) \neq \hat{f}_2(\mathbf{x}) \\ -1 & \hat{f}_1(\mathbf{x}) = \hat{f}_2(\mathbf{x}) \end{cases}$



$$\hat{f}_1(\mathbf{x}) = \text{sign}(\mathbf{w}_1^T \mathbf{x})$$

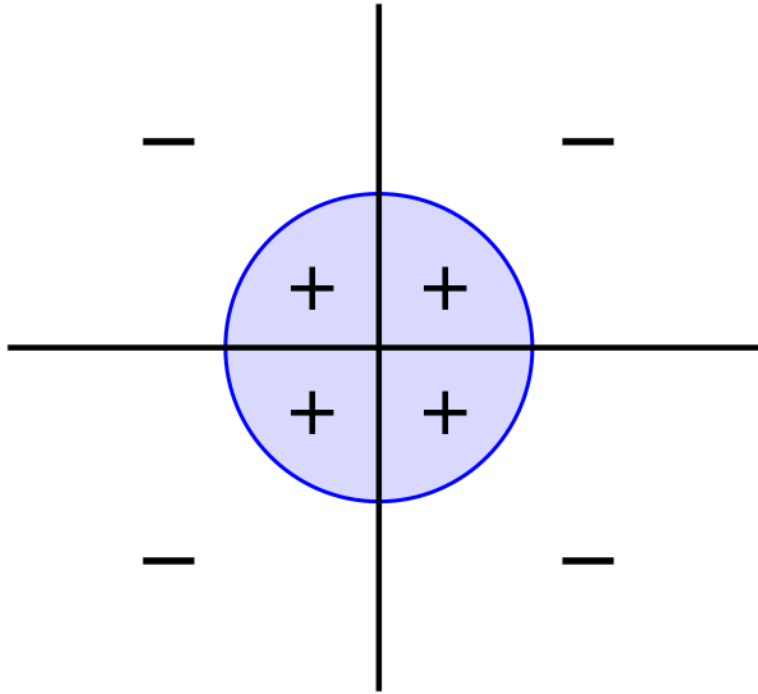


$$\hat{f}_2(\mathbf{x}) = \text{sign}(\mathbf{w}_2^T \mathbf{x})$$

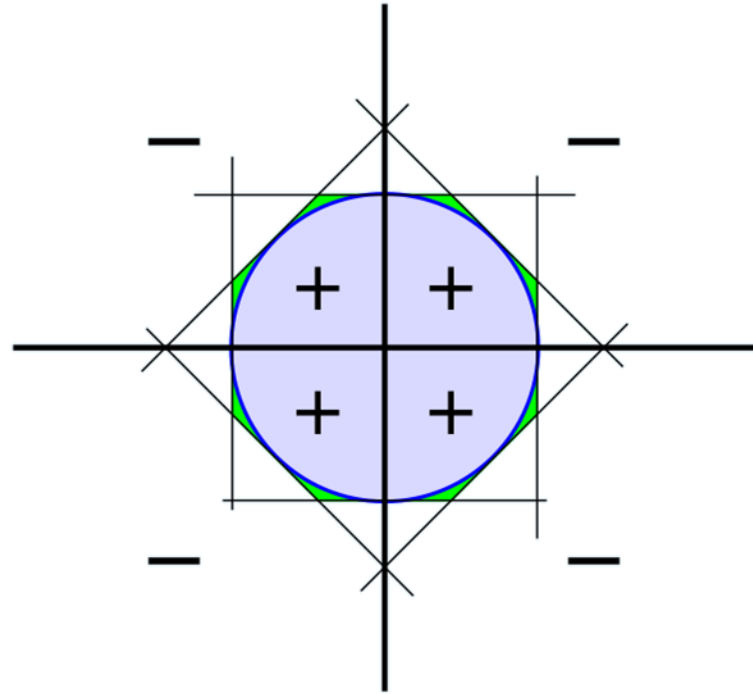


Source: Abu-Mostafa, Learning from Data, Caltech

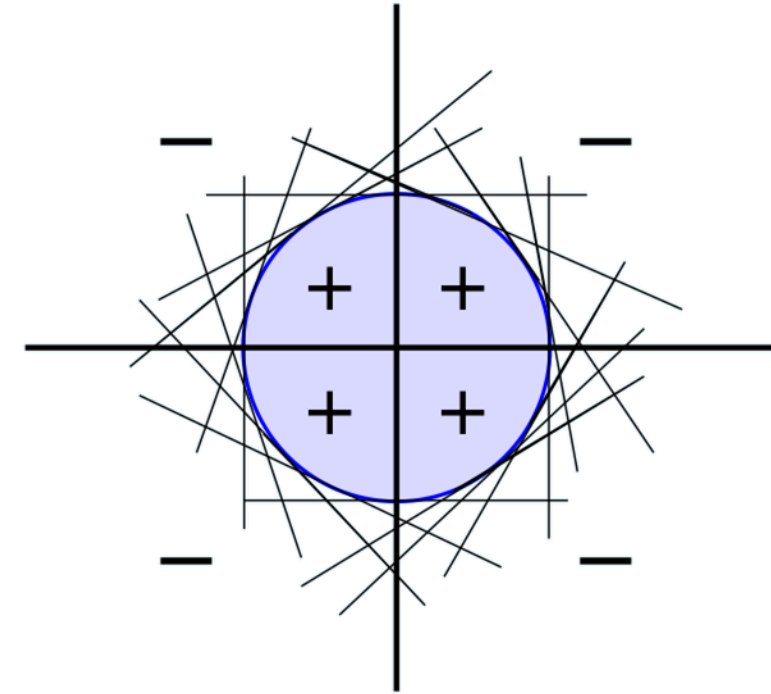
Multilayer Perceptron



Target



8 perceptrons



16 perceptrons

The more nodes/neurons, the more flexible is the model

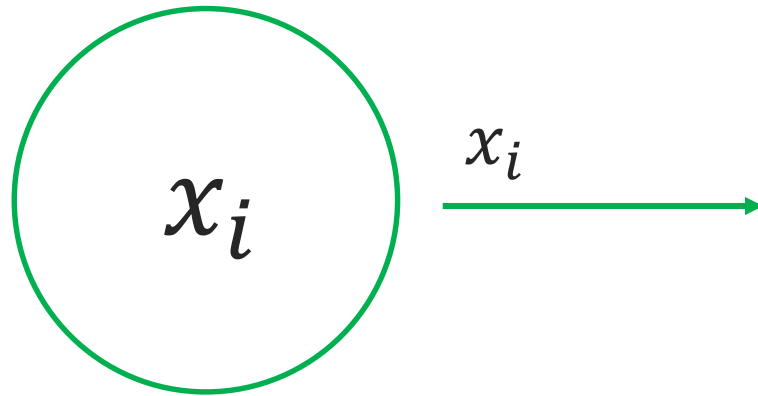
Source: Abu-Mostafa, Learning from Data, Caltech

Universal function approximation

“A **feedforward network** with a single layer is sufficient to represent **any function**, but the layer may be infeasibly large and may fail to learn and generalize correctly.”

Ian Goodfellow, Deep Learning
Creator of generative adversarial networks

Input nodes / neurons



Simply passes the input value to the next layer

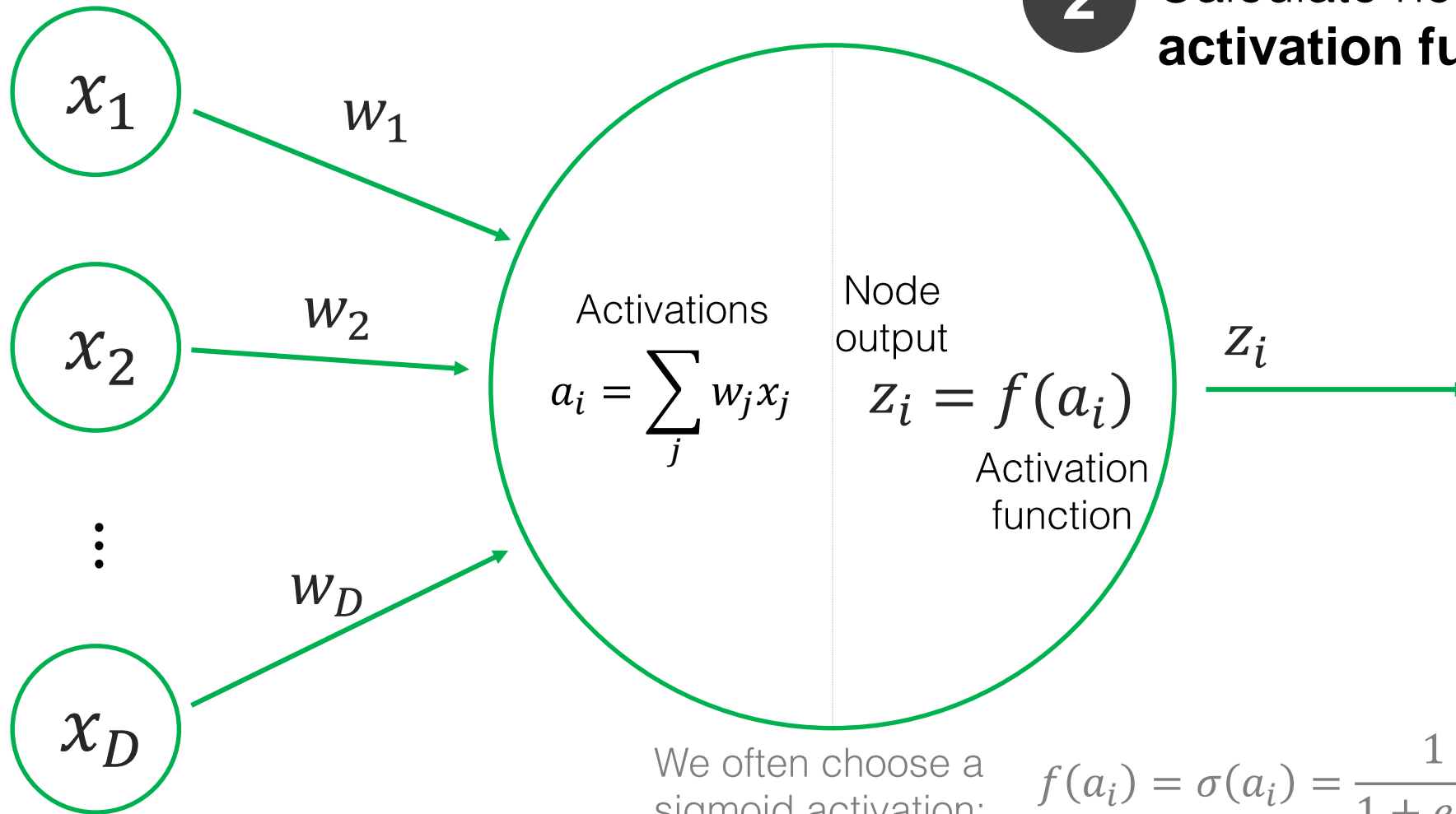
Hidden & output nodes

1

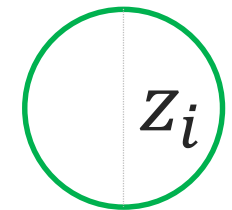
Calculate the **activations**: linear combinations of weights and the last layer's output

2

Calculate node output: apply the **activation function** to the activations



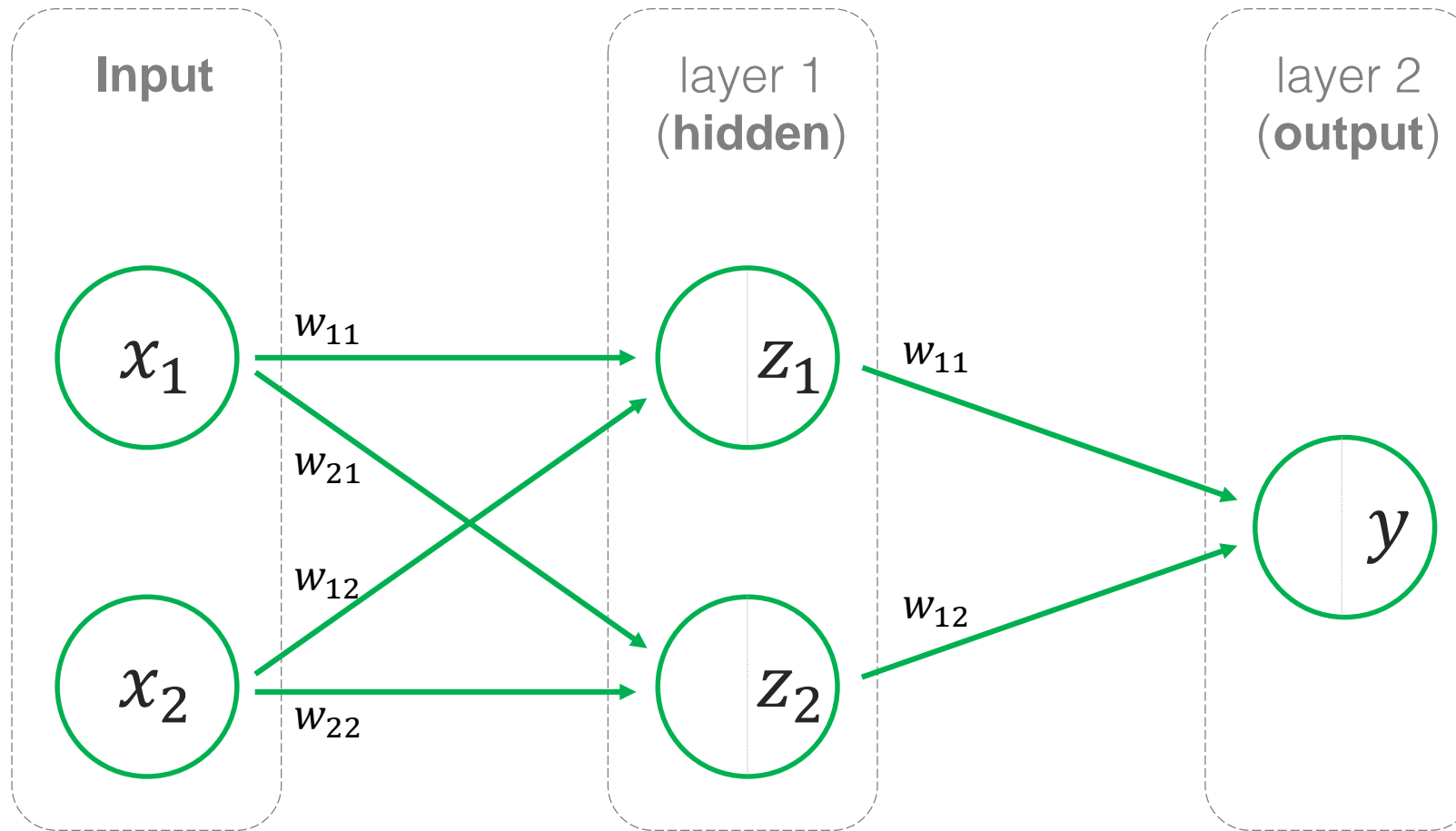
Represented as:



We often choose a sigmoid activation:

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

Simple Neural Network



Notational shorthand:
(a more precise
alternative notation)

$$w_{ij} = w_{ij}^{(1)}$$

$$z_i = z_i^{(1)}$$

$$w_{ij} = w_{ij}^{(2)}$$

$$y = z_1^{(2)}$$

$w_{ij}^{(k)}$

- Layer k
- From node j
(in the last layer)
- to node i
(in the next layer)

Forward Propagation

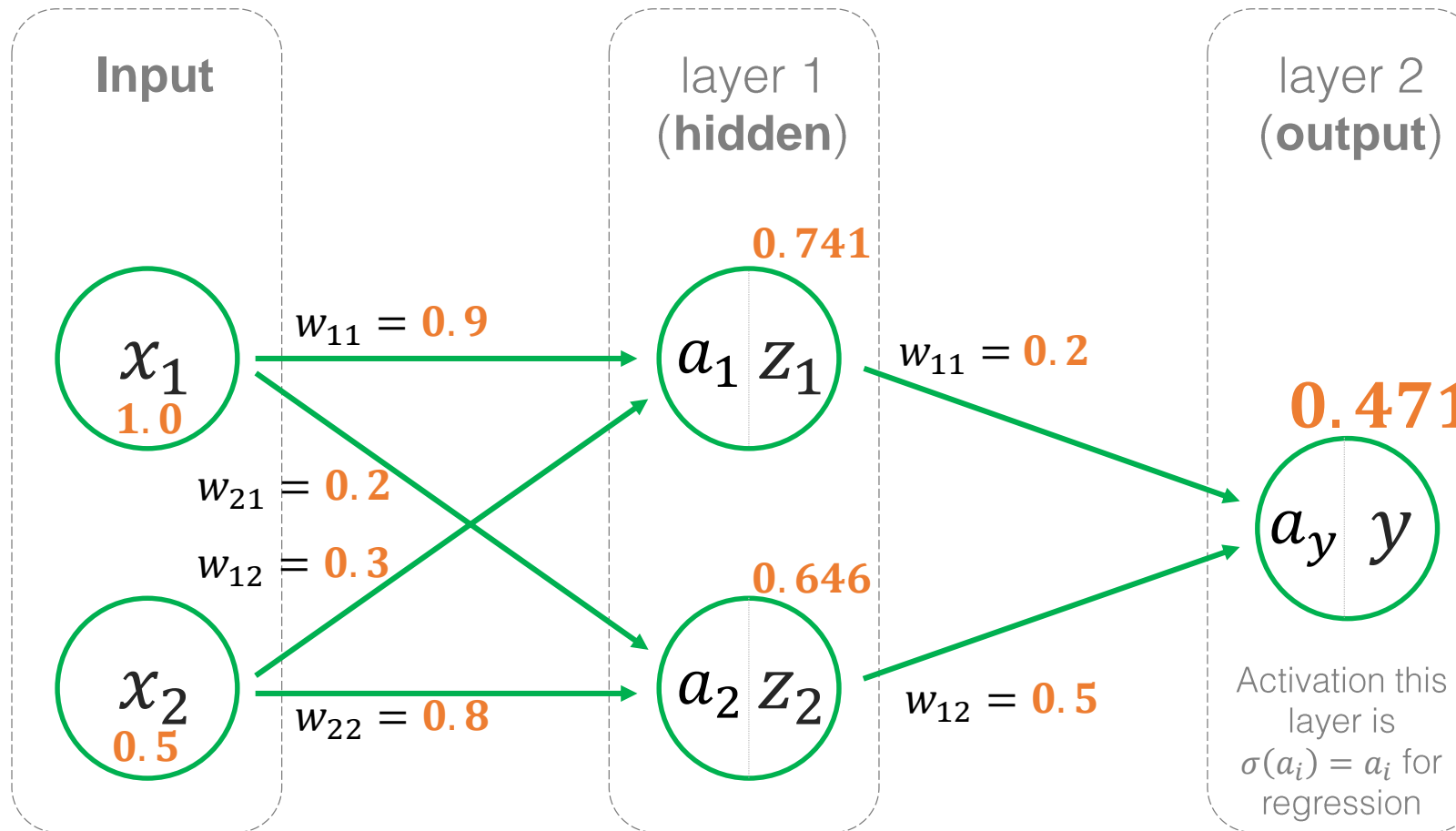
Calculating the output from input

$$a_1 = (0.9)(1.0) + (0.3)(0.5) = 1.05 \quad \text{Hidden layer calculations}$$

$$a_2 = (0.2)(1.0) + (0.8)(0.5) = 0.6$$

$$z_1 = \sigma(a_1) = \sigma(1.05) = 0.741$$

$$z_2 = \sigma(a_2) = \sigma(0.6) = 0.646$$



Output layer calculations

$$a_y = (0.2)(0.741) + (0.5)(0.646) = 0.471$$

$$y = a_y = 0.471 \quad \text{Regression}$$

Alternatively...

$$y = \sigma(a_y) = \sigma(0.471) = 0.616$$

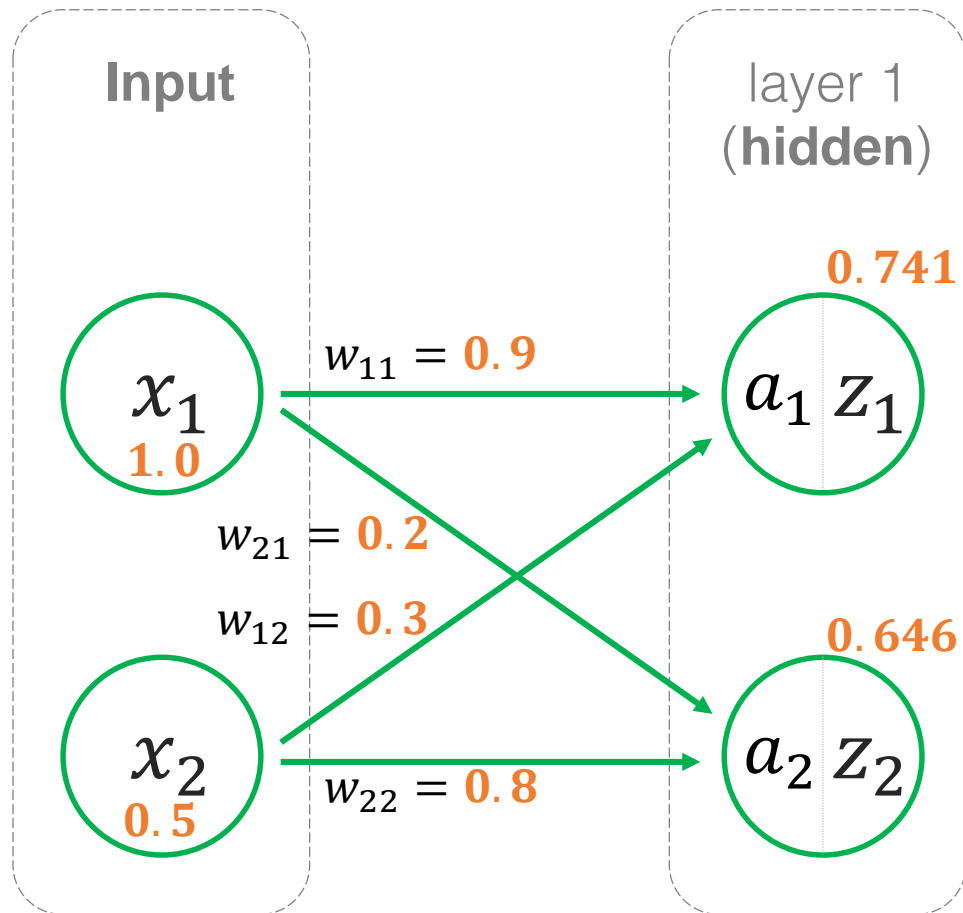
Classification

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

Activation this layer is $\sigma(a_i) = a_i$ for regression

Forward Propagation

Calculating the output from input



Hidden layer matrix calculations

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \quad \mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$$

$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} \begin{array}{l} \longrightarrow \text{The weights INTO node } z_1 \\ \longrightarrow \text{The weights INTO node } z_2 \end{array}$$

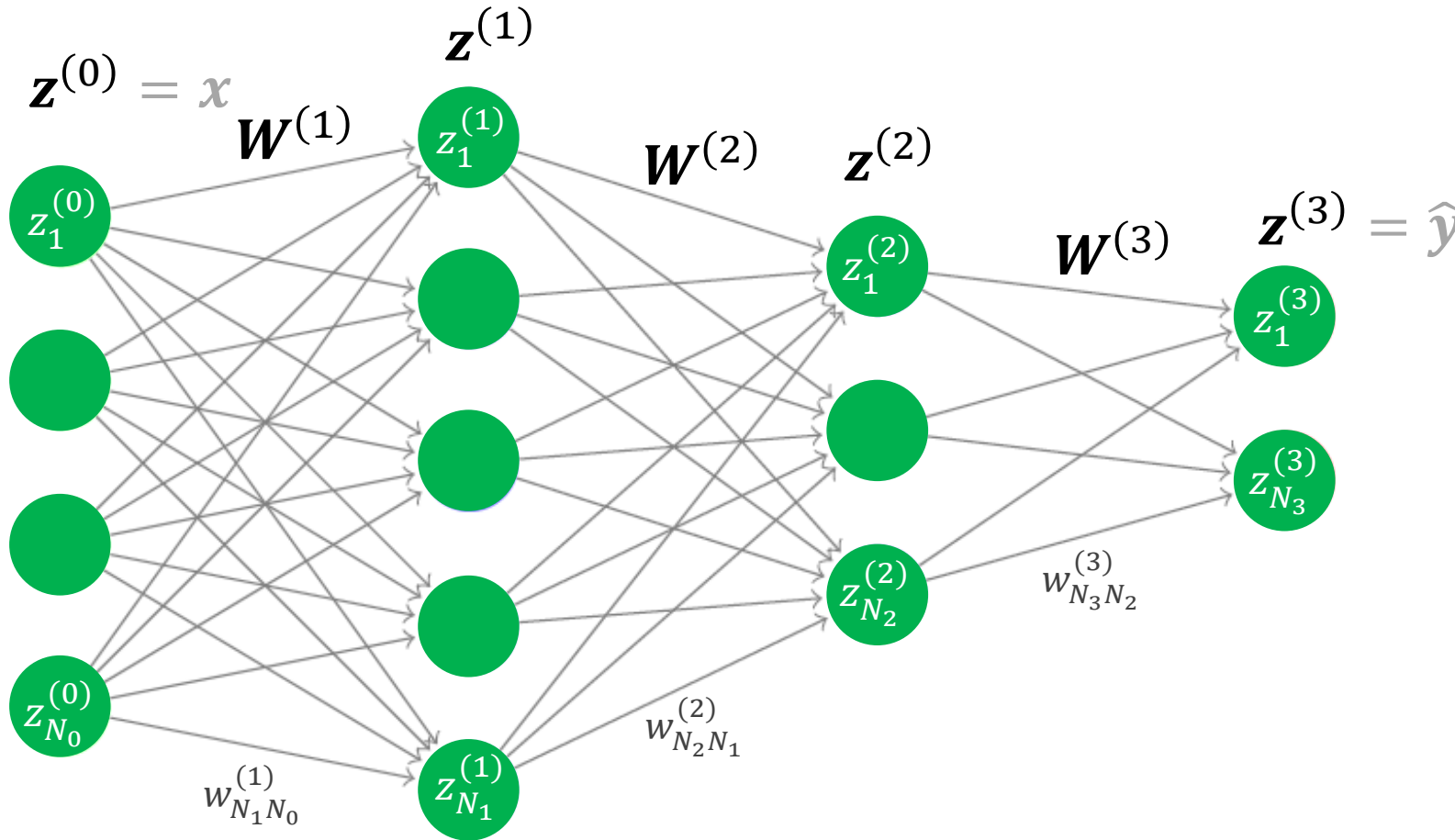
$$\mathbf{a} = \mathbf{W}\mathbf{x} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$= \begin{bmatrix} w_{11}x_1 + w_{12}x_2 \\ w_{21}x_1 + w_{22}x_2 \end{bmatrix}$$

$$\mathbf{z} = \sigma(\mathbf{a}) = \begin{bmatrix} \sigma(w_{11}x_1 + w_{12}x_2) \\ \sigma(w_{21}x_1 + w_{22}x_2) \end{bmatrix}$$

Forward Propagation

Example neural network with $L = 3$ layers and the i th layer has N_i nodes



Simple steps for forward propagation:

$$\text{For } i = 1 \text{ to } L - 1:$$
$$\mathbf{z}^{(i)} = \sigma(\mathbf{W}^{(i)} \mathbf{z}^{(i-1)})$$

Where:

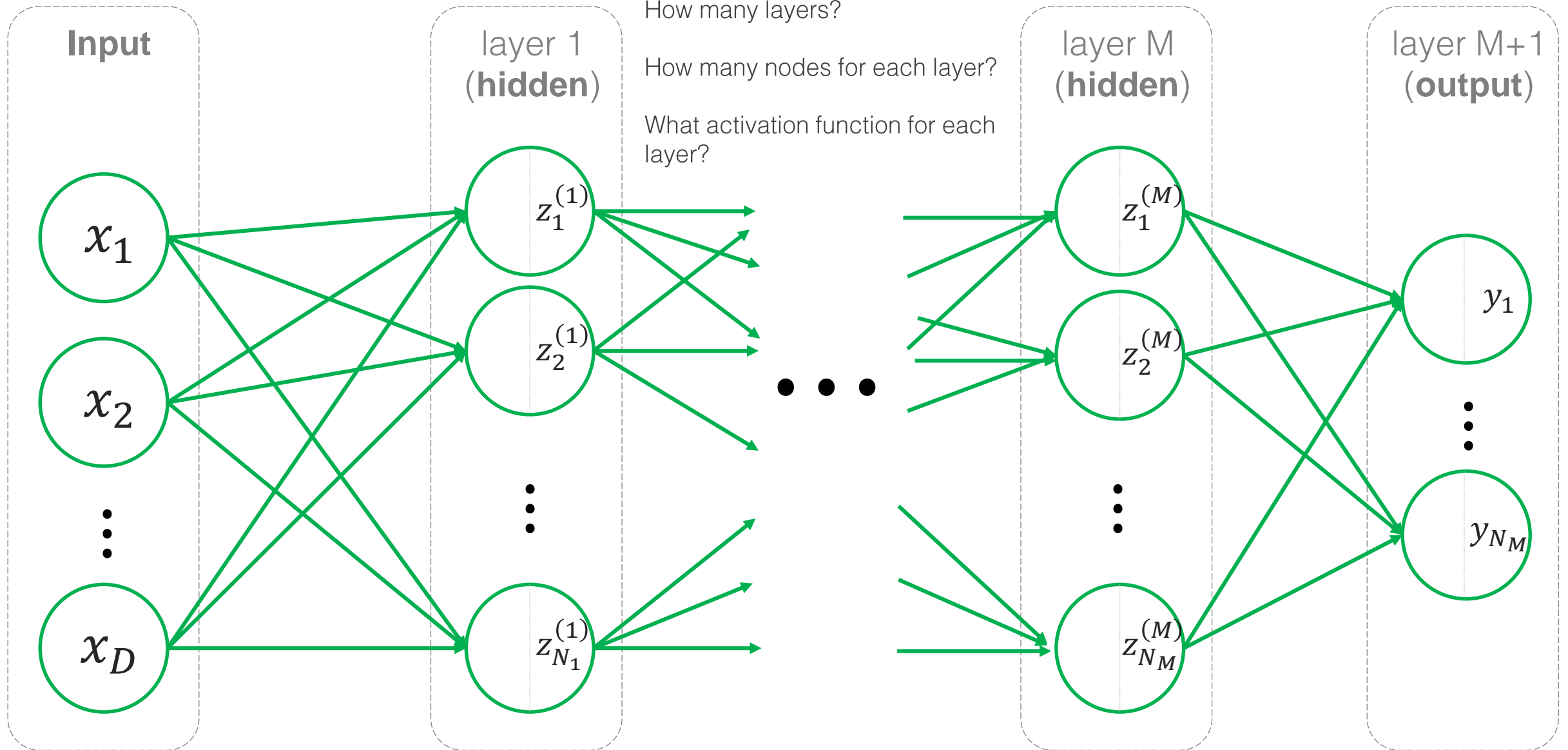
$$\mathbf{z}^{(0)} = \mathbf{x}$$

$$\hat{\mathbf{y}} = \mathbf{z}^{(L)}$$

Prediction error is measured:

$$E_n = \frac{1}{2} (\hat{y}_n - y_n)^2$$

Neural networks can be customized

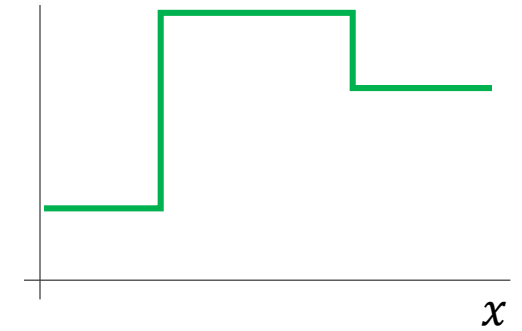
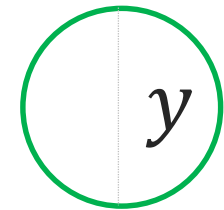
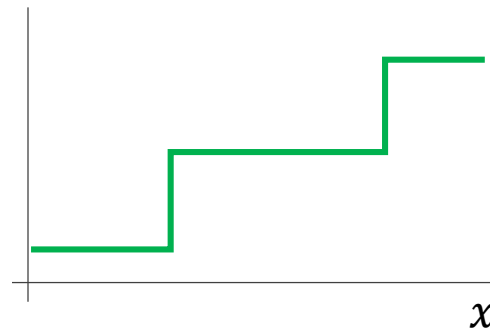
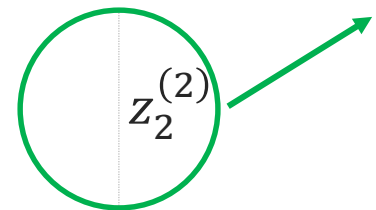
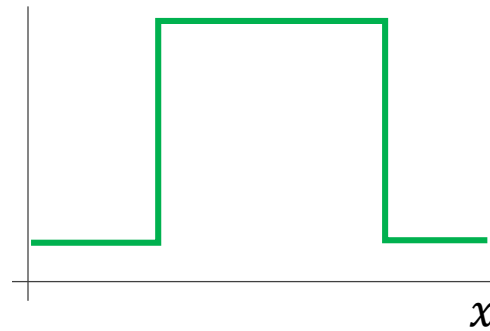
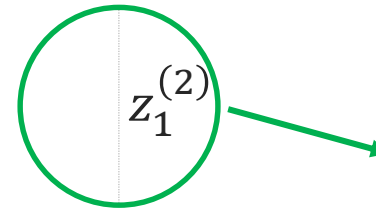
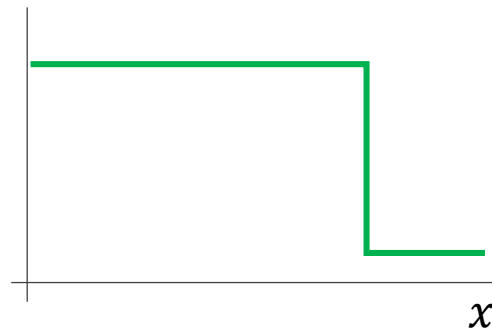
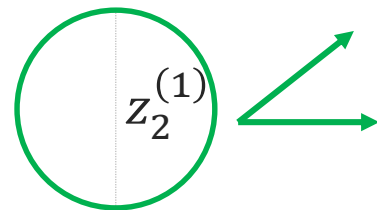
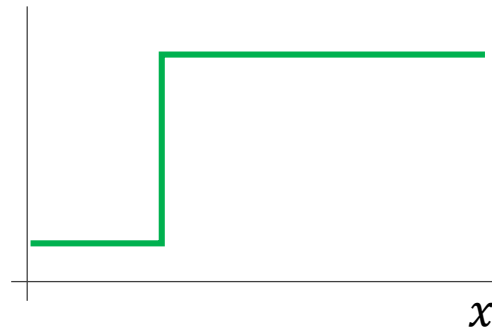
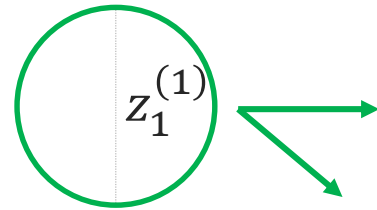
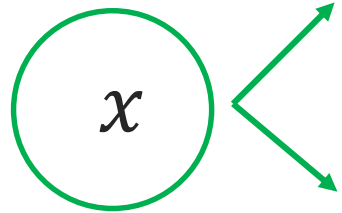


Input

Hidden 1

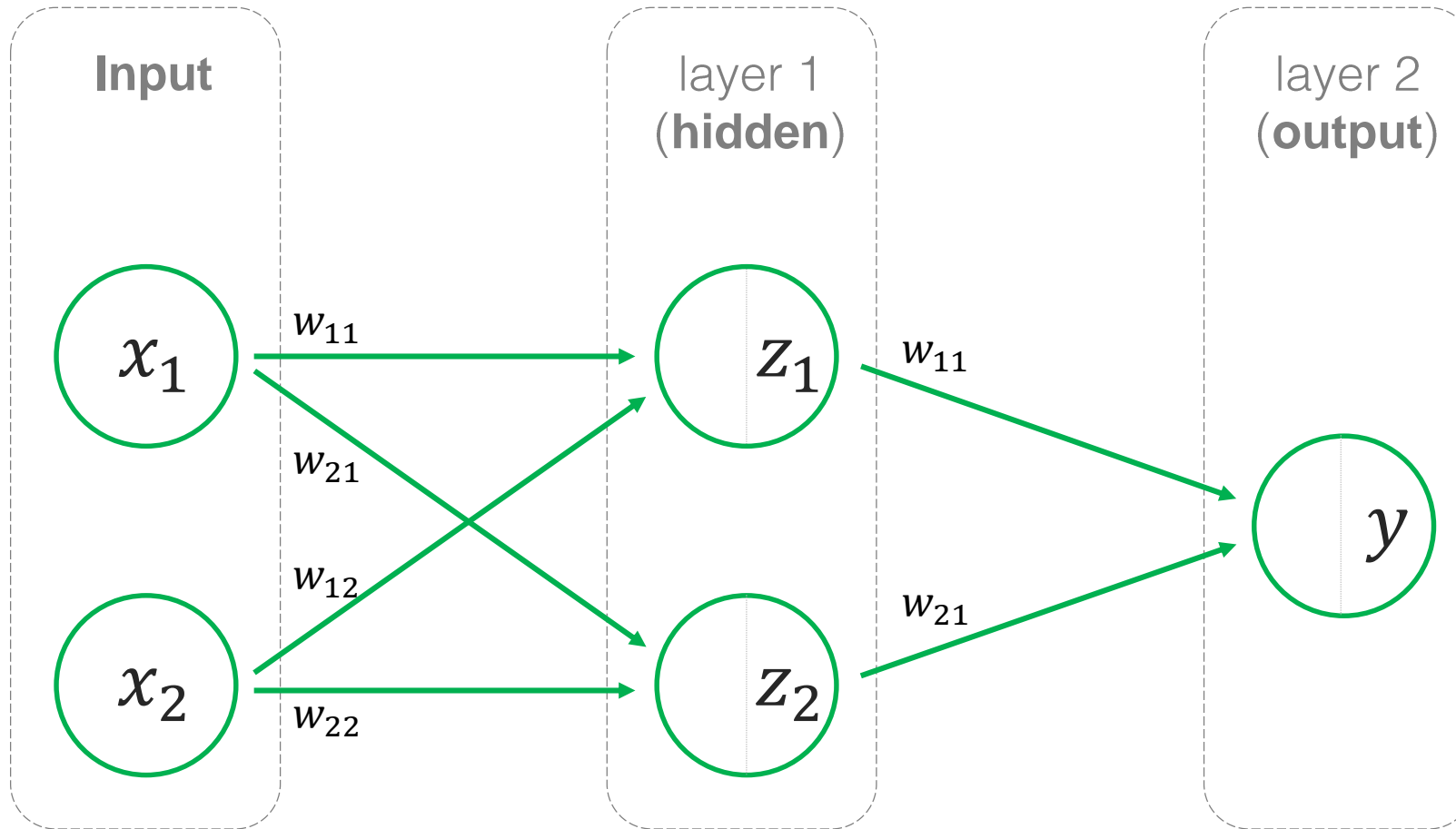
Hidden 2

Output



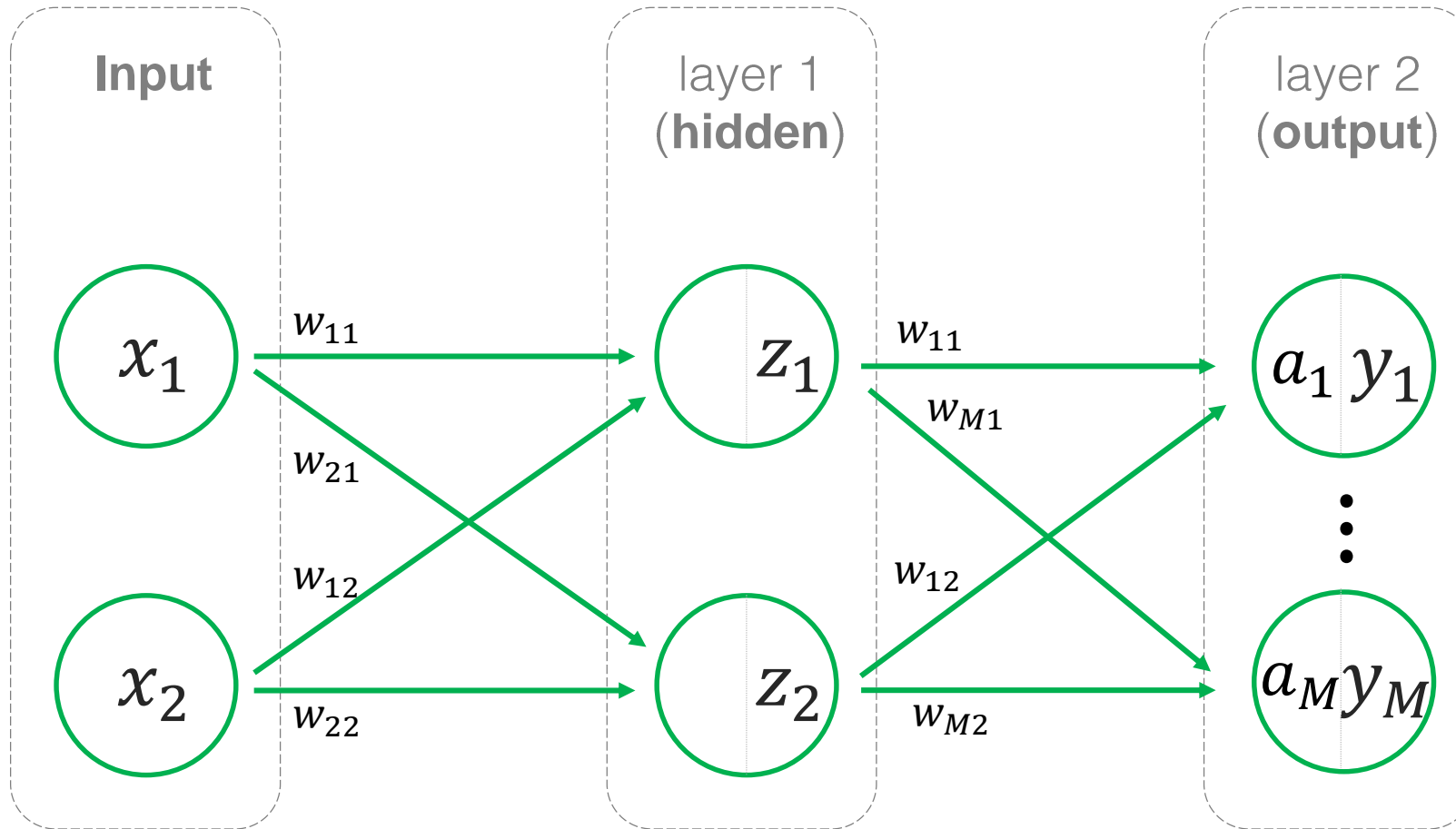
Multilayer neural nets can build up from basic building blocks to more complex structures

From binary to multiclass classification



For **binary classification** with a sigmoid activation function, the output is between zero and one, so threshold this value to assign the class

From binary to multiclass classification



For **multiclass problems**, we can have multiple outputs and use a softmax function:

$$y_i = g(a_i) = \frac{e^{a_i}}{\sum_{n=1}^M e^{a_n}}$$

Choose the largest y value as the predicted class

As with many aspects of neural networks this is one of a number of approaches

Next time...

What is a neural network and **how does it work?**

How do we **choose model weights?**

(i.e. how do we fit our model to data)

What are the challenges of using neural networks?