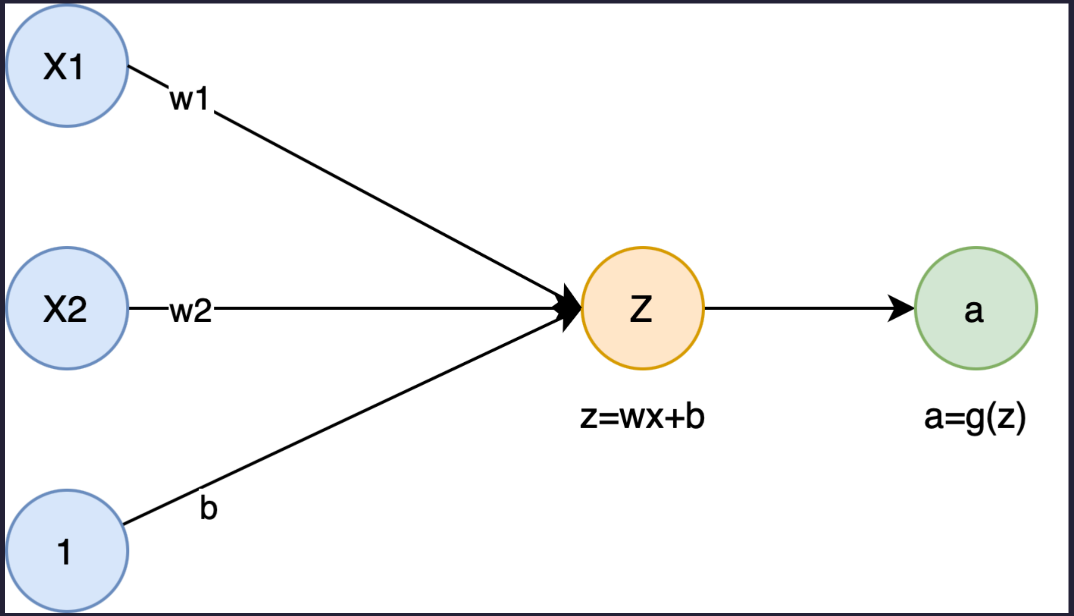
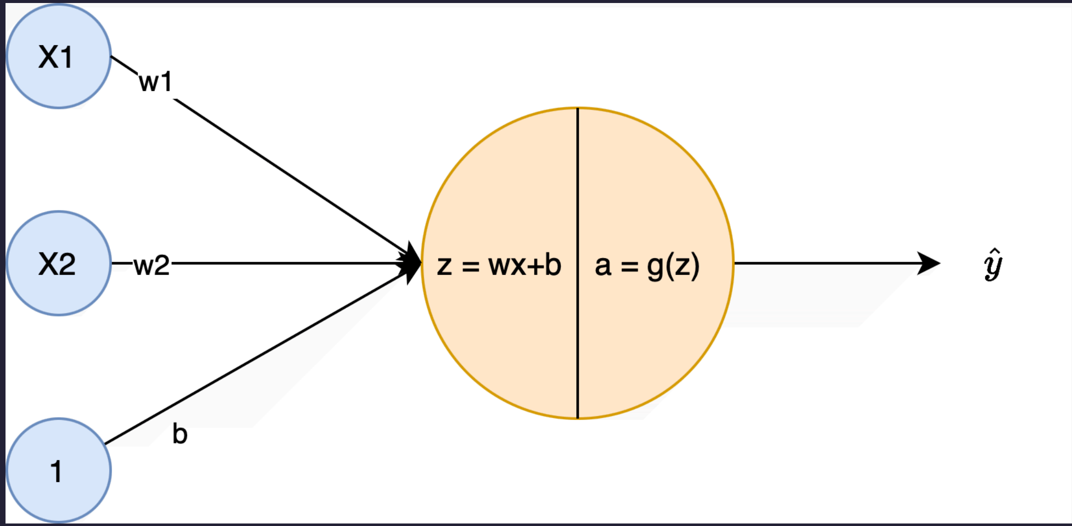
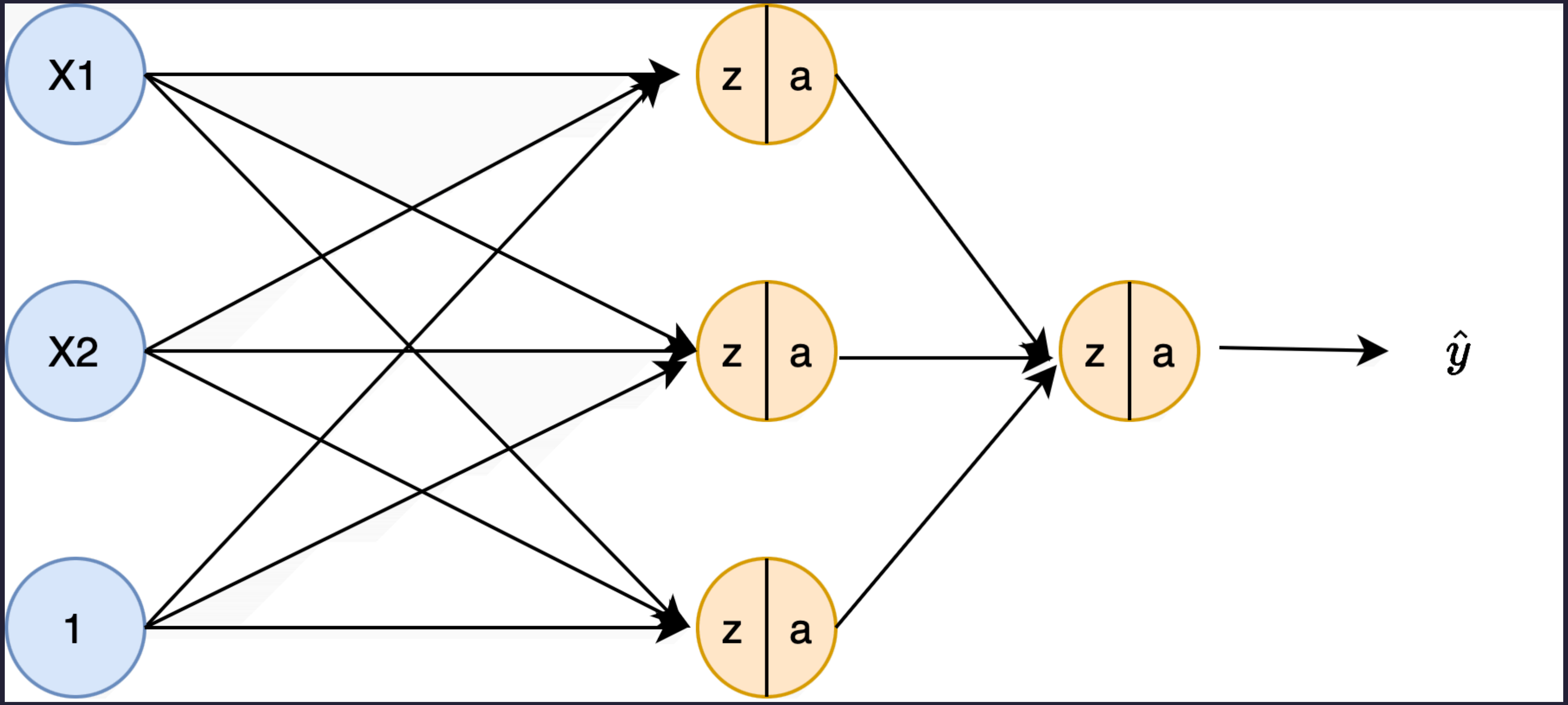
Neural networks

Recap: Logistic regression

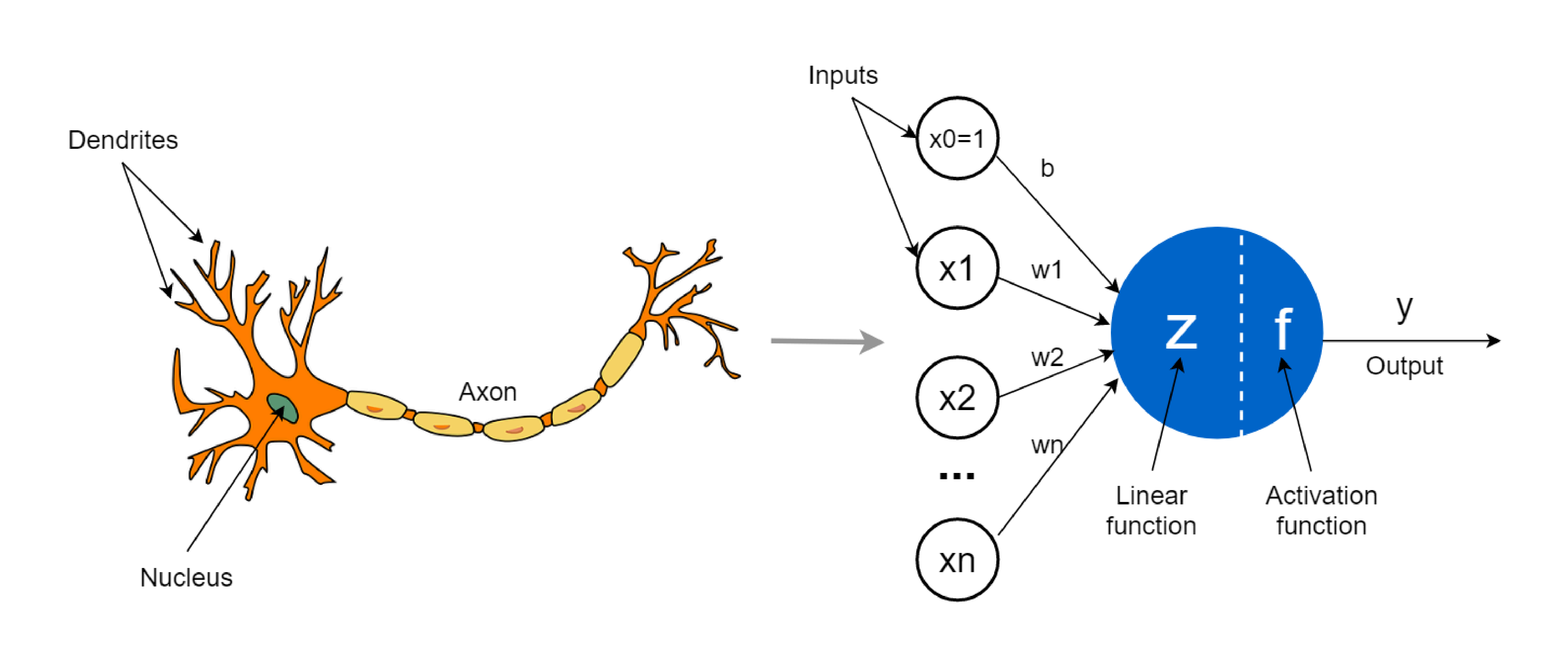




What is a neural network?



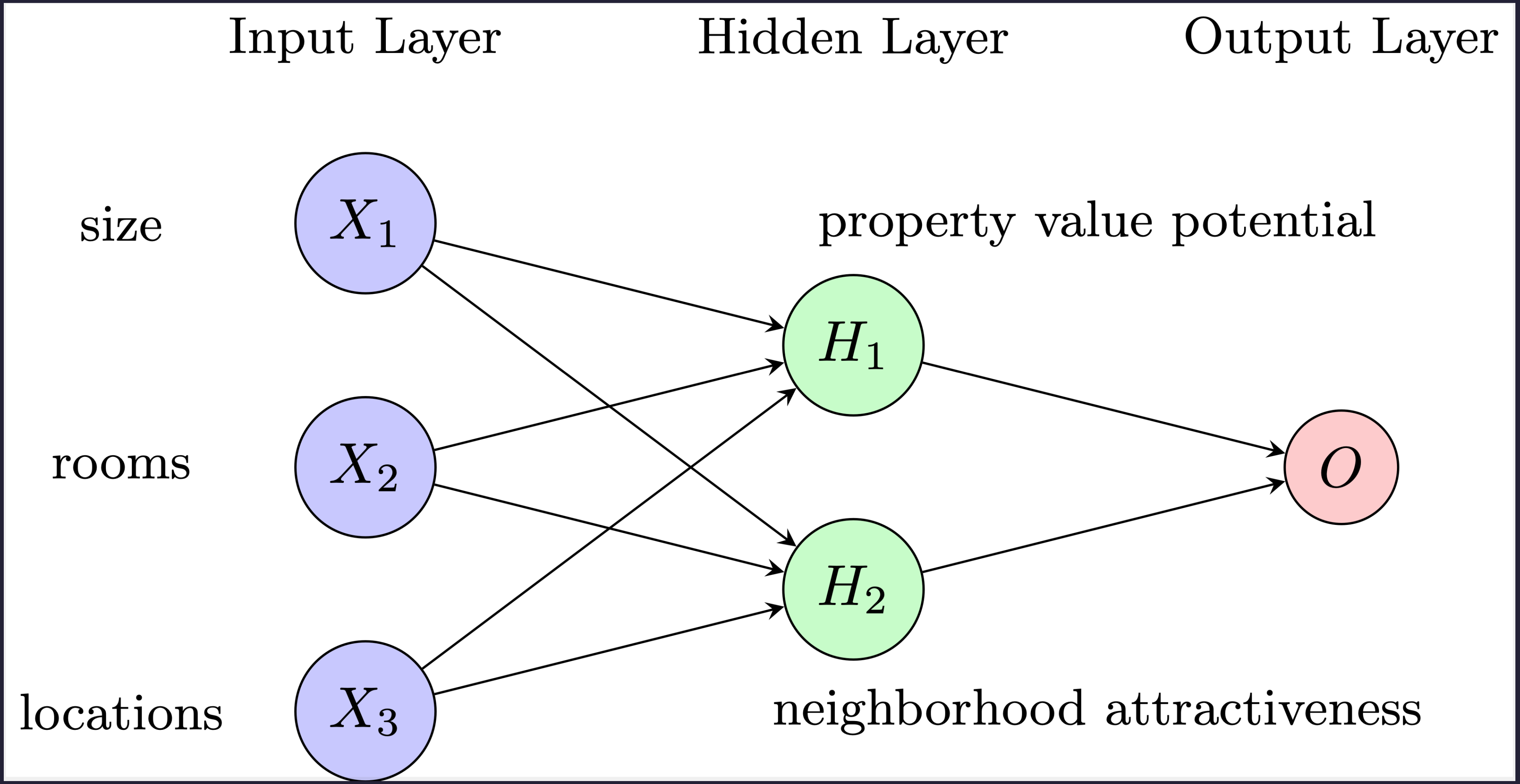
Simplified model of a biological neuron



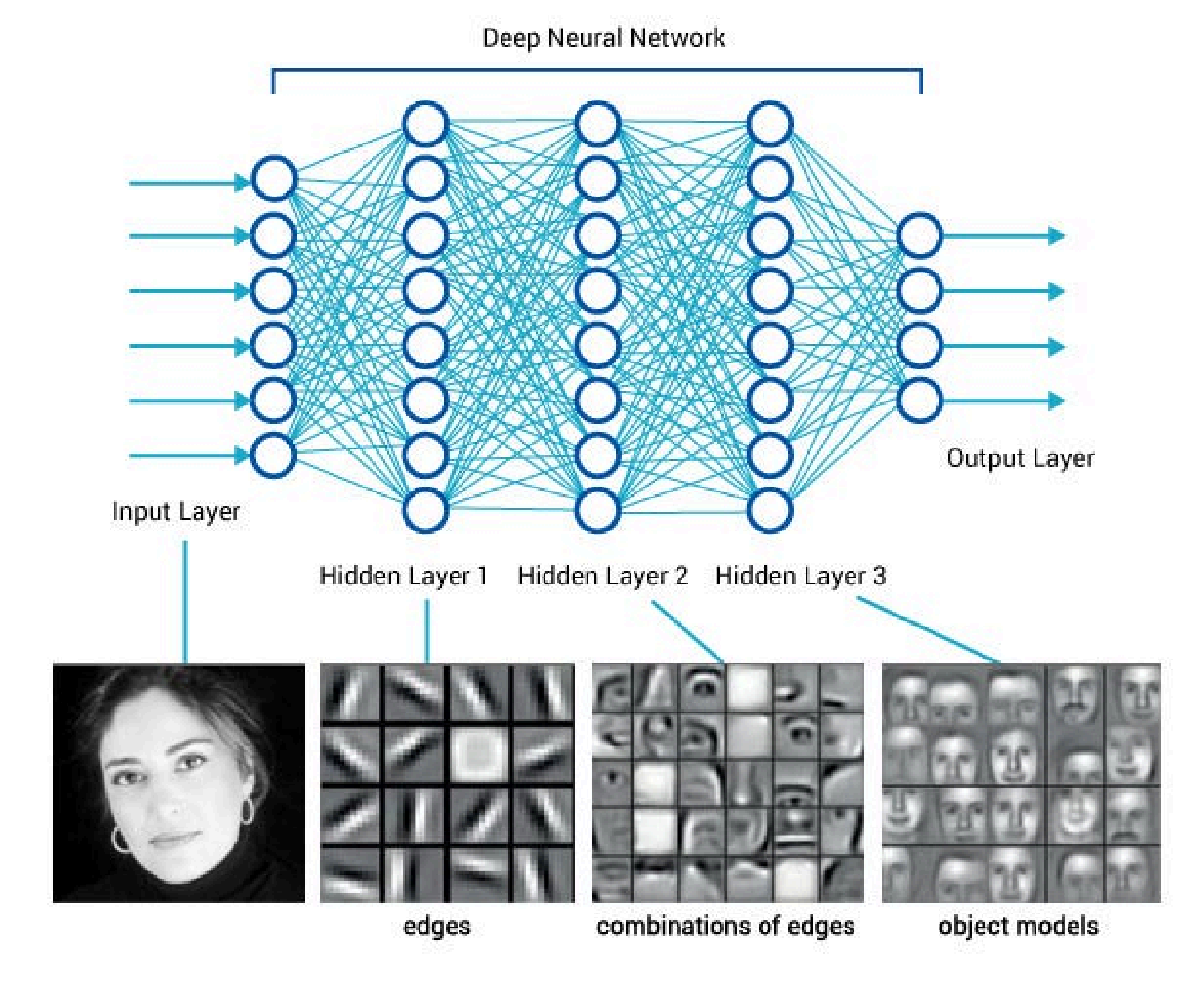
Why "deep" representations?

- **Feature extraction**: each layer learns a new representation of the input
- **Hierarchical representation**: each layer learns a more abstract representation of the input

Feature extraction



Hierarchical representation



Shallow neural networks

Model representation - Input layer

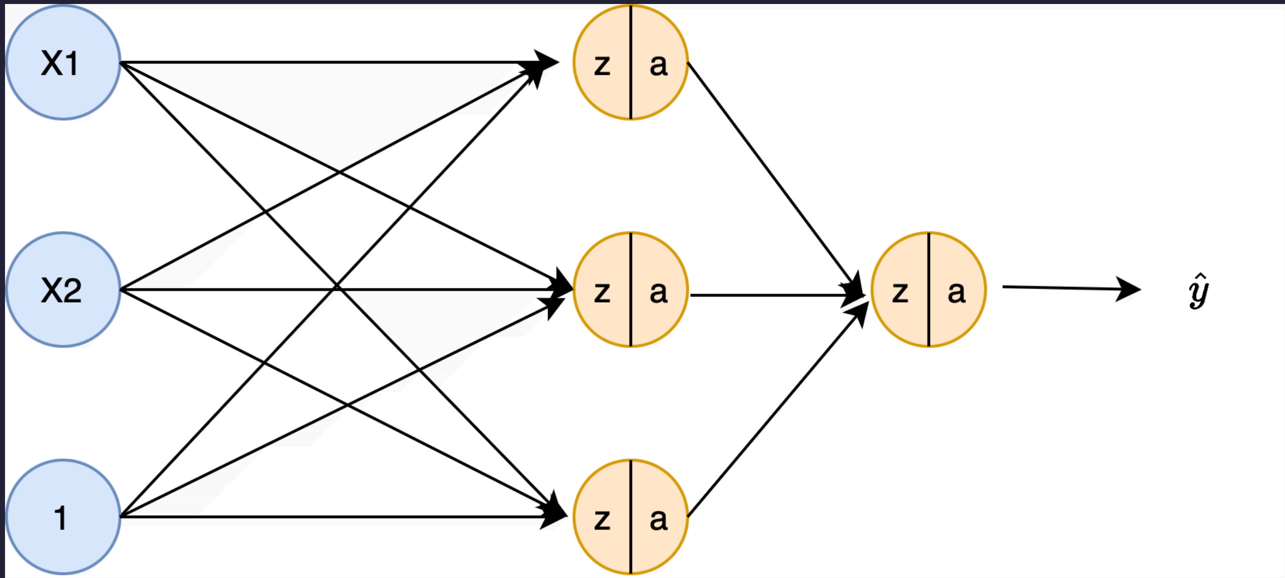
Layer 0: 2 units ($n_x = 2$)

single training example ($n_m = 1$)

- $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (n_x, n_m)$

m training examples ($n_m = m$)

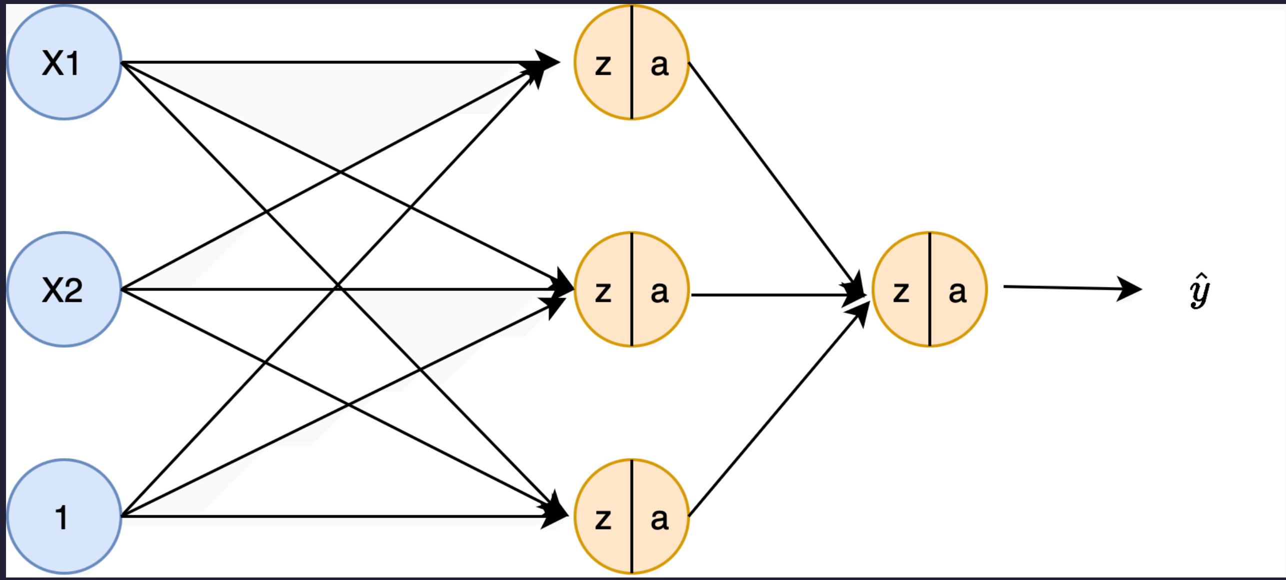
- $X = \begin{bmatrix} x_1^{(1)} & x_1^{(2)} & \cdots & x_1^{(m)} \\ x_2^{(1)} & x_2^{(2)} & \cdots & x_2^{(m)} \end{bmatrix}$
- (n_x, n_m)



Model representation - Hidden layer

Layer 1: 3 units ($n_h = 3$)

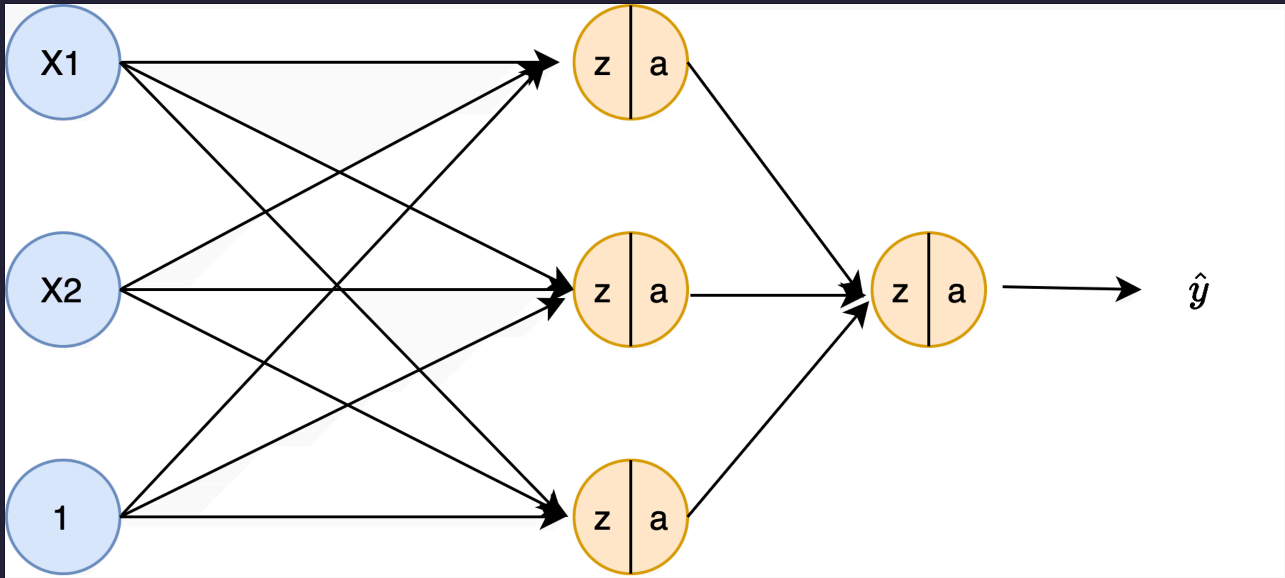
- $W^{[1]} = \begin{bmatrix} w_{11}^{[1]} & w_{12}^{[1]} \\ w_{21}^{[1]} & w_{22}^{[1]} \\ w_{31}^{[1]} & w_{32}^{[1]} \end{bmatrix} \quad (n_h, n_x)$
- $b^{[1]} = \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \end{bmatrix} \quad (n_h, 1)$
- $z^{[1]} = W^{[1]}x + b^{[1]} \quad (n_h, n_m)$
- $a^{[1]} = g^{[1]}(z^{[1]}) \quad (n_h, n_m)$



Model representation - Output layer

Layer 2: 1 unit ($n_y = 1$)

- $W^{[2]} = \begin{bmatrix} w_{11}^{[2]} & w_{12}^{[2]} & w_{13}^{[2]} \end{bmatrix} \quad (n_y, n_h)$
- $b^{[2]} = b^{[2]} \quad (n_y, 1)$
- $z^{[2]} = W^{[2]} a^{[1]} + b^{[2]} \quad (n_h, n_m)$
- $a^{[2]} = g^{[2]}(z^{[2]}) \quad (n_y, n_m)$

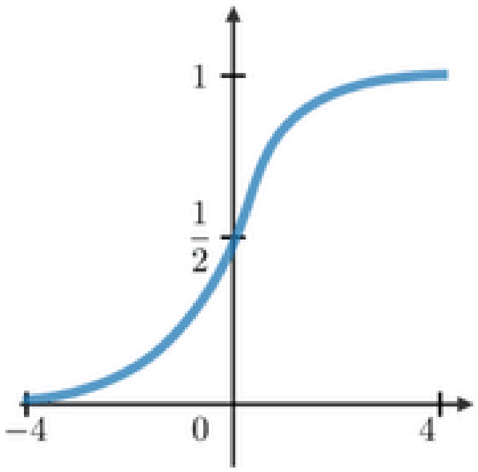
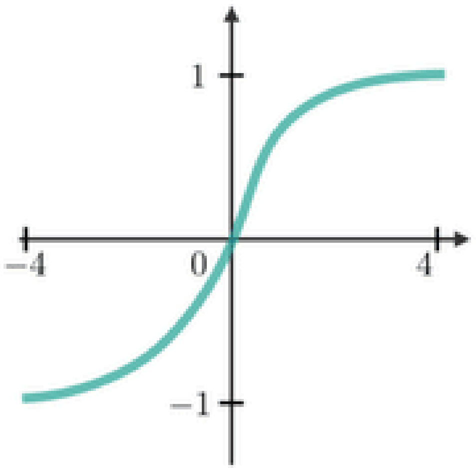
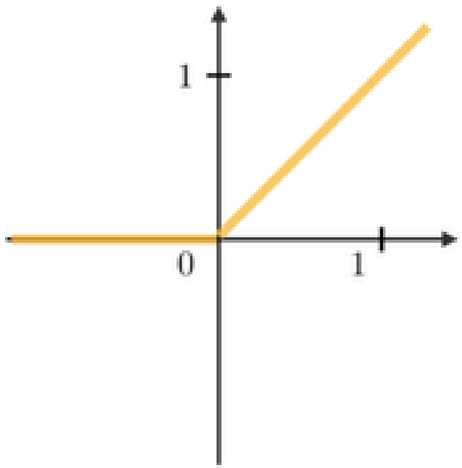
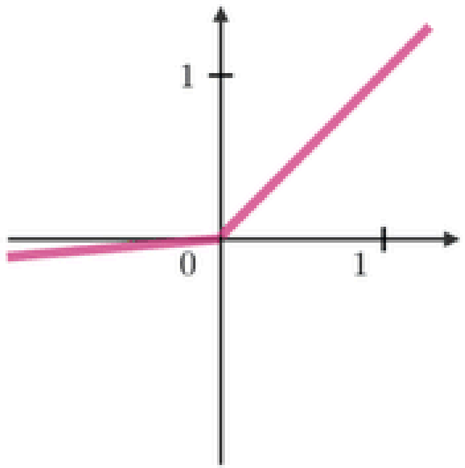


Activation functions

$$a = g(z)$$

- Each neuron applies an activation function to the net input z
- the activation function introduces non-linearity to the model

Activation functions

Sigmoid	Tanh	ReLU	Leaky ReLU
$g(z) = \dfrac{1}{1+e^{-z}}$	$g(z) = \dfrac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$	$g(z) = \max(0, z)$	$g(z) = \max(\epsilon z, z)$ with $\epsilon \ll 1$
			

Which activation function to use?

Hidden layer:

- Sigmoid: non-zero centered, vanishing gradient ➡️ not recommended
- Tanh: zero-centered, vanishing gradient ➡️ better than sigmoid
- **ReLU**: most common

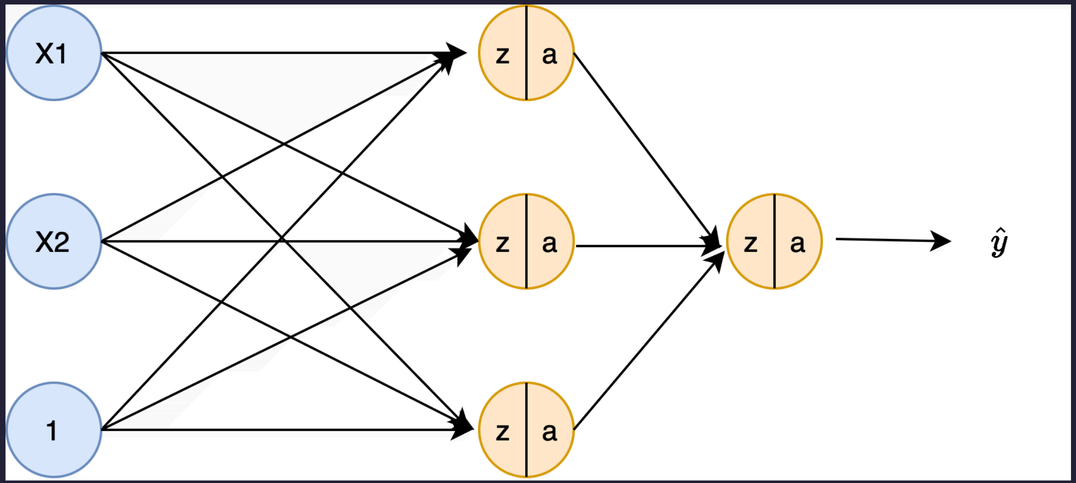
Output layer:

- Sigmoid: binary classification
- Softmax: multi-class classification
- Linear: regression

Why non-linear activation functions?

$$z^{[1]} = W^{[1]}x + b^{[1]}$$
$$a^{[1]} = \underbrace{g(z^{[1]}) = z^{[1]}}_{\text{linear activation}}$$
$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$
$$a^{[2]} = \underbrace{g(z^{[2]}) = z^{[2]}}_{\text{linear activation}}$$

$$\begin{aligned} a^{[2]} &= W^{[2]}(W^{[1]}x + b^{[1]}) + b^{[2]} \\ &= W^{[2]}W^{[1]}x + W^{[2]}b^{[1]} + b^{[2]} \\ &= W'x + b' \end{aligned}$$



Training Neural Networks

Forward Propagation:

- computes the output of the neural network given an input by passing the input through each layer
- the final prediction for that input

Backward Propagation:

- computes the gradients of the cost function with respect to the parameters by propagating the error backward through the network
- how each parameter affects the cost function, enabling the optimization algorithm to adjust them to minimize the error

Forward Propagation ➡️

input: x

activation function: $g(z) = 1/(1 + e^{-z})$

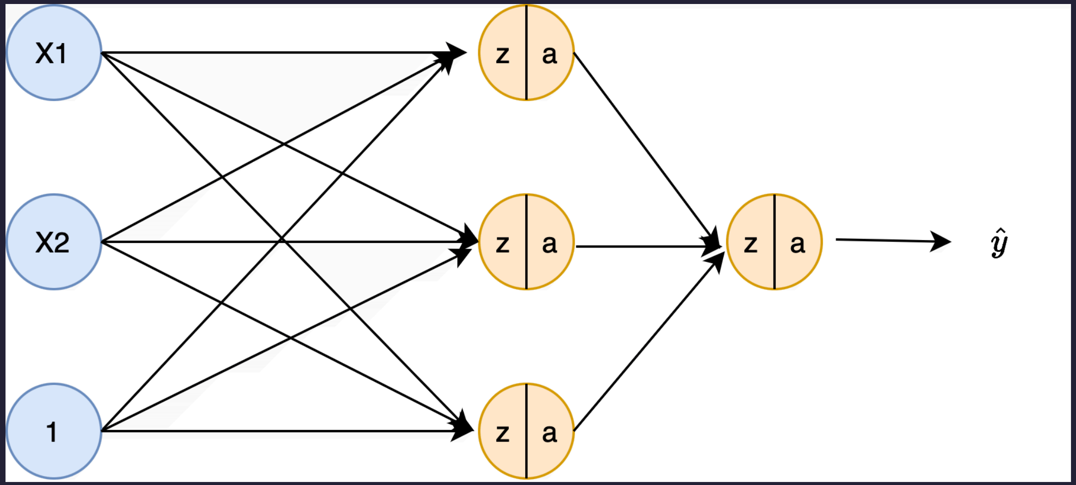
$$z^{[1]} = W^{[1]}x + b^{[1]}$$

$$a^{[1]} = g(z^{[1]})$$

$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$

$$a^{[2]} = g(z^{[2]})$$

output: $a^{[2]}$



Gradient Descent for Neural Networks

Parameters: $W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}$

Cost function: $\frac{1}{m} L(a^{[2]}, y)$

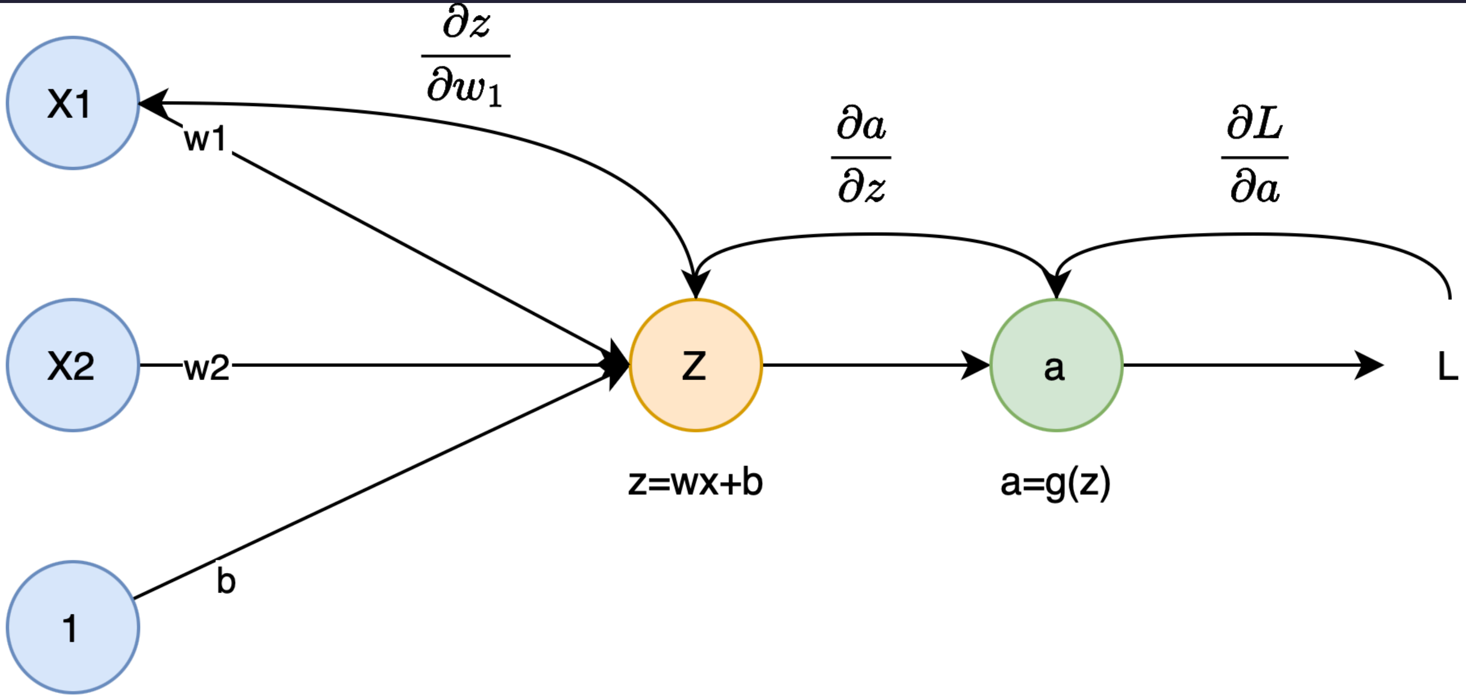
Repeat until convergence

$$W^{[2]} = W^{[2]} - \alpha \frac{\partial L}{\partial W^{[2]}}$$

$$b^{[2]} = b^{[2]} - \alpha \frac{\partial L}{\partial b^{[2]}}$$

$$W^{[1]} = W^{[1]} - \alpha \frac{\partial L}{\partial W^{[1]}}$$

$$b^{[1]} = b^{[1]} - \alpha \frac{\partial L}{\partial b^{[1]}}$$



$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial a}\frac{\partial a}{\partial z}\frac{\partial z}{\partial w_1}$$

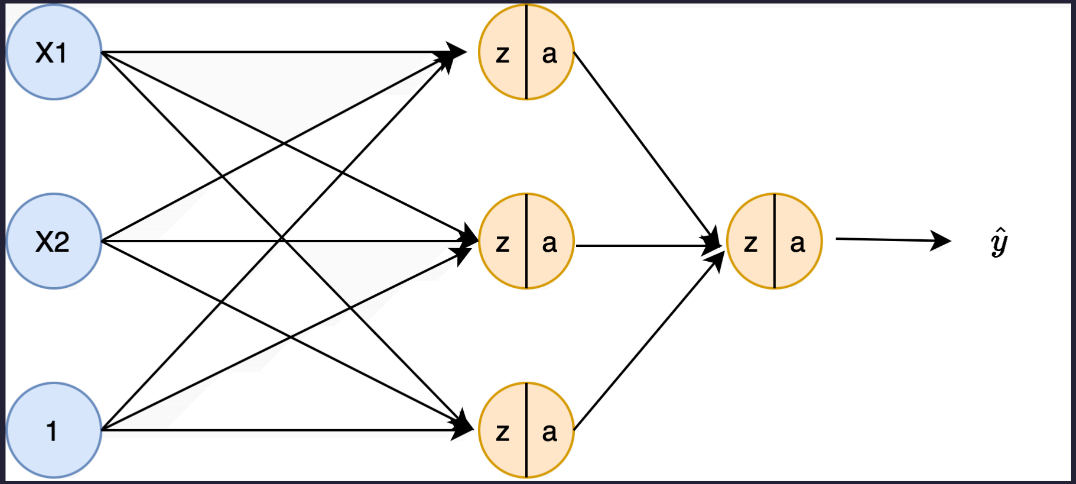
Backward Propagation ⬅️

Layer 2:

$$\frac{\partial L}{\partial z^{[2]}} = \frac{\partial L}{\partial a^{[2]}} \frac{\partial a^{[2]}}{\partial z^{[2]}}$$

$$\frac{\partial L}{\partial W^{[2]}} = \frac{\partial L}{\partial z^{[2]}} \frac{\partial z^{[2]}}{\partial W^{[2]}}$$

$$\frac{\partial L}{\partial b^{[2]}} = \frac{\partial L}{\partial z^{[2]}} \frac{\partial z^{[2]}}{\partial b^{[2]}}$$



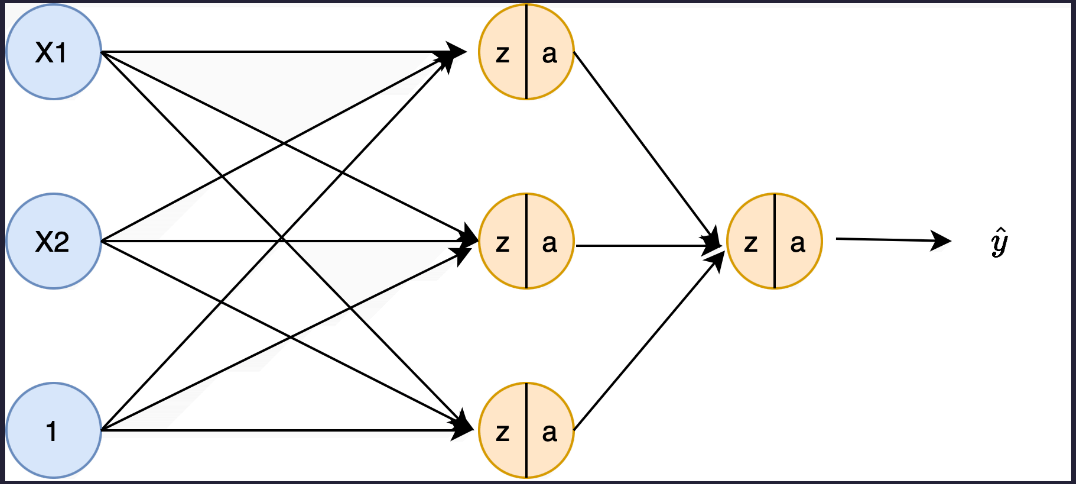
Backward Propagation ⬅️

Layer 1:

$$\frac{\partial L}{\partial z^{[1]}} = \frac{\partial L}{\partial z^{[2]}} \frac{\partial z^{[2]}}{\partial a^{[1]}} \frac{\partial a^{[1]}}{\partial z^{[1]}}$$

$$\frac{\partial L}{\partial W^{[1]}} = \frac{\partial L}{\partial z^{[1]}} \frac{\partial z^{[1]}}{\partial W^{[1]}}$$

$$\frac{\partial L}{\partial b^{[1]}} = \frac{\partial L}{\partial z^{[1]}} \frac{\partial z^{[1]}}{\partial b^{[1]}}$$



Backprop for binary classification - Layer 2

$$L(a^{[2]}, y) = -y\log(a^{[2]}) - (1-y)\log(1-a^{[2]})$$
$$a^{[2]} = g(z^{[2]}) = 1/(1+e^{-z^{[2]}})$$

Layer 2:

$$\frac{\partial L}{\partial z^{[2]}} = \frac{\partial L}{\partial a^{[2]}}\frac{\partial a^{[2]}}{\partial z^{[2]}}$$
$$\frac{\partial L}{\partial W^{[2]}} = \frac{\partial L}{\partial z^{[2]}}\frac{\partial z^{[2]}}{\partial W^{[2]}}$$
$$\frac{\partial L}{\partial b^{[2]}} = \frac{\partial L}{\partial z^{[2]}}\frac{\partial z^{[2]}}{\partial b^{[2]}}$$

Layer 2:

$$dz^{[2]} = (\tfrac{a^{[2]}-y}{a^{[2]}(1-a^{[2]})})a^{[2]}(1-a^{[2]}) = a^{[2]} - y$$
$$dW^{[2]} = \tfrac{1}{m}dz^{[2]}a^{[1]T}$$
$$db^{[2]} = \tfrac{1}{m}dz^{[2]}$$

See Logistic Regression slides for the derivations

Backprop for binary classification - Layer 1

$$L(a^{[2]}, y) = -y\log(a^{[2]}) - (1-y)\log(1-a^{[2]})$$
$$a^{[1]} = g(z^{[1]}) = 1/(1+e^{-z^{[1]}})$$

Layer 1:

$$\frac{\partial L}{\partial z^{[1]}} = \frac{\partial L}{\partial z^{[2]}}\frac{\partial z^{[2]}}{\partial a^{[1]}}\frac{\partial a^{[1]}}{\partial z^{[1]}}$$
$$\frac{\partial L}{\partial W^{[1]}} = \frac{\partial L}{\partial z^{[1]}}\frac{\partial z^{[1]}}{\partial W^{[1]}}$$
$$\frac{\partial L}{\partial b^{[1]}} = \frac{\partial L}{\partial z^{[1]}}\frac{\partial z^{[1]}}{\partial b^{[1]}}$$

Layer 1:

$$dz^{[1]} = W^{[2]T}dz^{[2]} * (a^{[1]}(1-a^{[1]}))$$
$$dW^{[1]} = \frac{1}{m}dz^{[1]}x^T$$
$$db^{[1]} = \frac{1}{m}dz^{[1]}$$

Formulas for binary classification

$$L(a^{[2]}, y) = -y\log(a^{[2]}) - (1-y)\log(1-a^{[2]})$$
$$a^{[2]} = g(z^{[2]}) = 1/(1+e^{-z^{[2]}})$$

Forward Propagation ➡️:

$$z^{[1]} = W^{[1]}x + b^{[1]}$$

$$a^{[1]} = g(z^{[1]})$$

$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$

$$a^{[2]} = g(z^{[2]})$$

Backward Propagation ⬅️:

$$dz^{[2]} = a^{[2]} - y$$

$$dW^{[2]} = \frac{1}{m}dz^{[2]}a^{[1]T}$$

$$db^{[2]} = \frac{1}{m}dz^{[2]}$$

$$dz^{[1]} = W^{[2]T}dz^{[2]} * (a^{[1]}(1-a^{[1]}))$$

$$dW^{[1]} = \frac{1}{m}dz^{[1]}x^T$$

$$db^{[1]} = \frac{1}{m}dz^{[1]}$$

Formulas for binary classification

$$L(a^{[2]}, y) = -y\log(a^{[2]}) - (1-y)\log(1-a^{[2]})$$
$$a^{[2]} = g(z^{[2]}) = 1/(1+e^{-z^{[2]}})$$

Forward Propagation ➡️:

$$z^{[1]} = W^{[1]}x + b^{[1]}$$

$$a^{[1]} = g(z^{[1]})$$

$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$

$$a^{[2]} = g(z^{[2]})$$

output: $a^{[2]}$

cache: $z^{[1]}, a^{[1]}, z^{[2]}, a^{[2]}$

Backward Propagation ⬅️:

$$dz^{[2]} = a^{[2]} - y$$

$$dW^{[2]} = \frac{1}{m} dz^{[2]} a^{[1]T}$$

$$db^{[2]} = \frac{1}{m} dz^{[2]}$$

$$dz^{[1]} = W^{[2]T} dz^{[2]} * (a^{[1]}(1 - a^{[1]}))$$

$$dW^{[1]} = \frac{1}{m} dz^{[1]} x^T$$

$$db^{[1]} = \frac{1}{m} dz^{[1]}$$

output: $dW^{[1]}, db^{[1]}, dW^{[2]}, db^{[2]}$

Formulas for regression

$$L(a^{[2]}, y) = \frac{1}{2}(a^{[2]} - y)^2$$
$$a^{[2]} = z^{[2]}$$

Forward Propagation ➡️:

$$z^{[1]} = W^{[1]}x + b^{[1]}$$

$$a^{[1]} = g(z^{[1]})$$

$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$

$$***a^{[2]} = z^{[2]}$$

Backward Propagation ⬅️:

$$dz^{[2]} = a^{[2]} - y$$

$$dW^{[2]} = \frac{1}{m}dz^{[2]}a^{[1]T}$$

$$db^{[2]} = \frac{1}{m}dz^{[2]}$$

$$dz^{[1]} = W^{[2]T}dz^{[2]} * (a^{[1]}(1 - a^{[1]}))$$

$$dW^{[1]} = \frac{1}{m}dz^{[1]}x^T$$

$$db^{[1]} = \frac{1}{m}dz^{[1]}$$

Random Initialization

Set weights to zero ➡️ Symmetry problem

- All units in the hidden layer will compute the same output
- No gradient direction, no learning

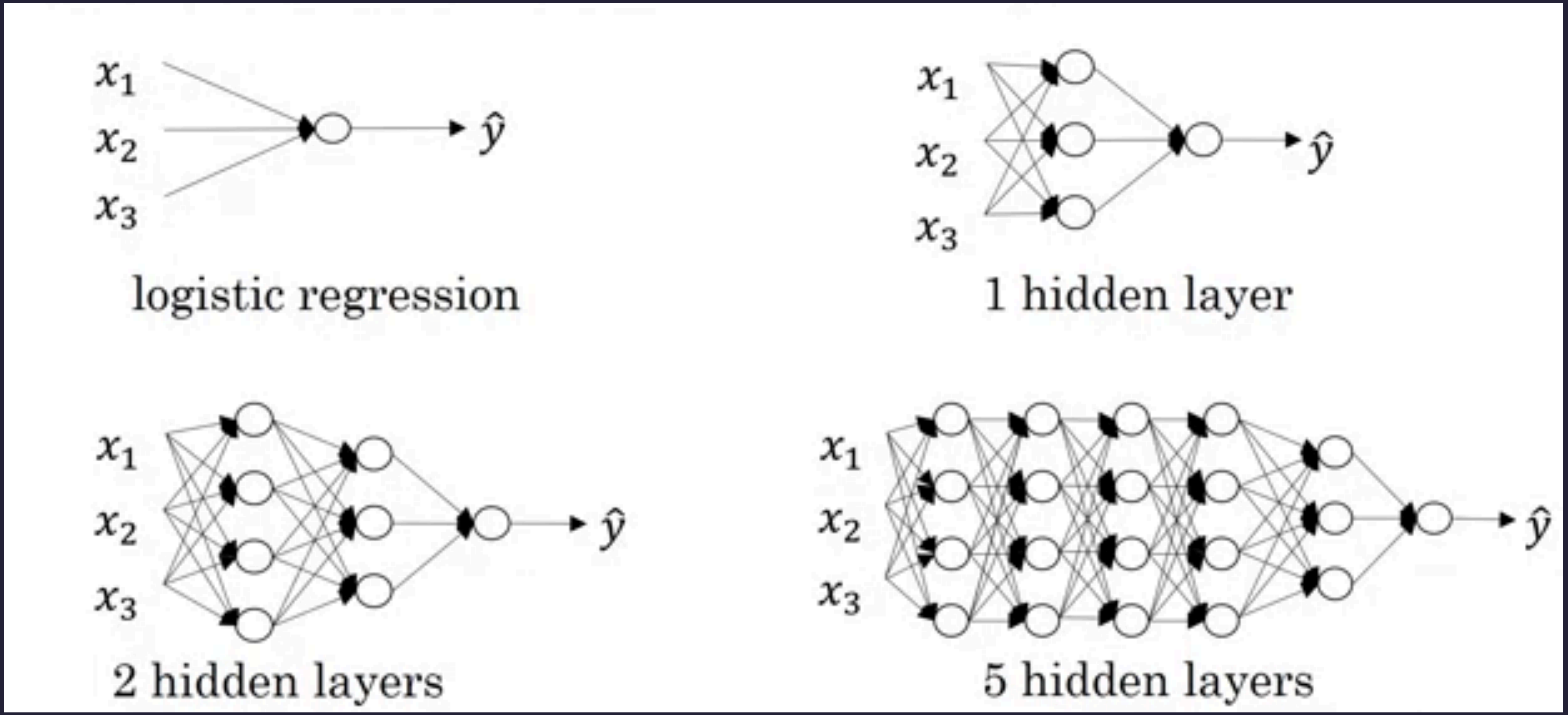
Set weights to random values instead

- Break symmetry
- Bias b can be set to zero.

```
W = np.random.randn((n_neurons, n_features)) * 0.01
b = np.zeros((n_neurons, 1))
```

Non-neural network models can be trained with zero initialization

Deep L-layer neural network



Layer types

Dense layer:

- Each neuron is connected to all neurons in the previous layer
- Multilayer Perceptron

Convolutional layer:

- Each neuron is connected to a local region of the previous layer
- Convolutional Neural Network (CNN)

Recurrent layer:

- Each neuron is connected to both the previous layer and its previous state, allowing it to retain information across time steps
- LSTM, GRU

💻 Multilayer Perceptron (MLP)