CSE 575
Statistical Machine Learning

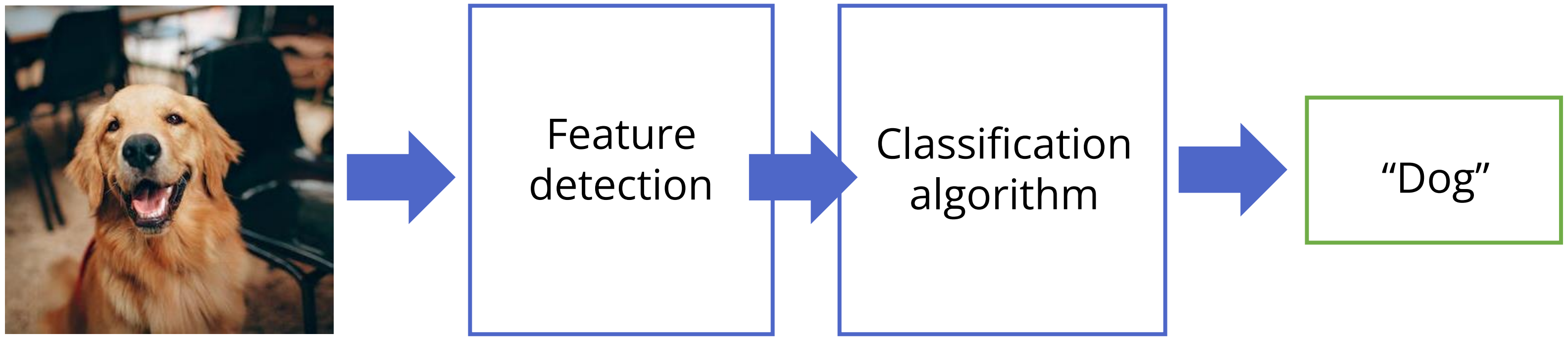
Lecture 12

YooJung Choi

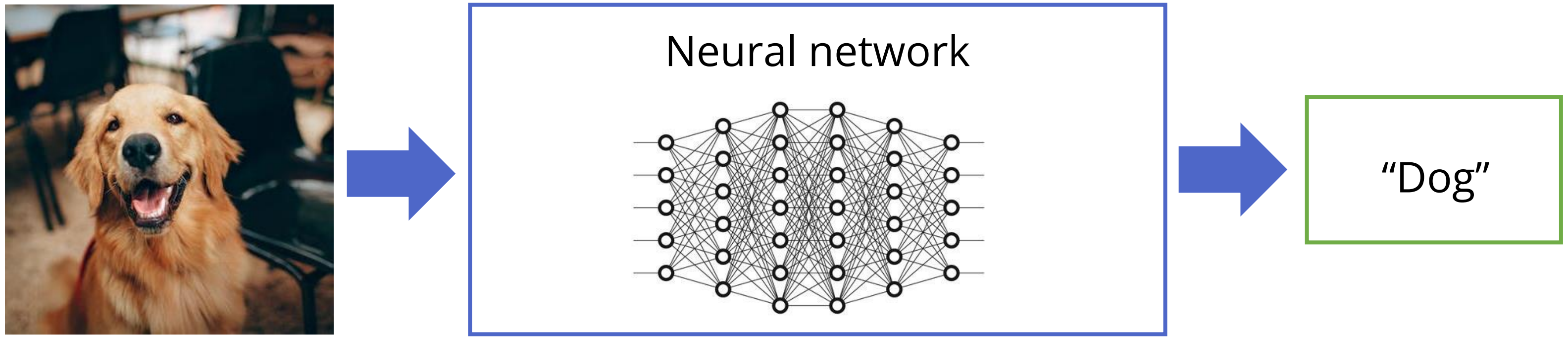
Fall 2022

Deep learning

Classical ML:
1. Determine features
2. Feed them through an ML model



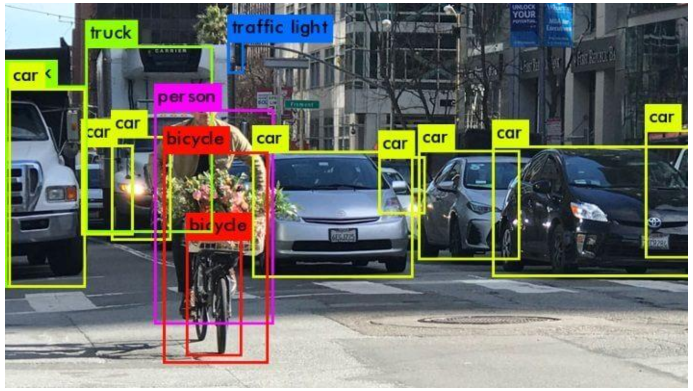
Deep learning:
2 steps combined into 1 task



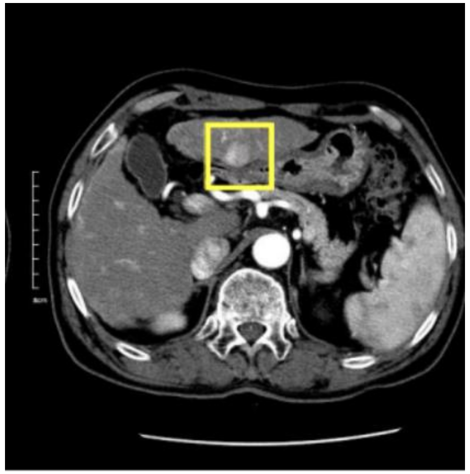
Deep learning applications

Computer vision

Facial recognition



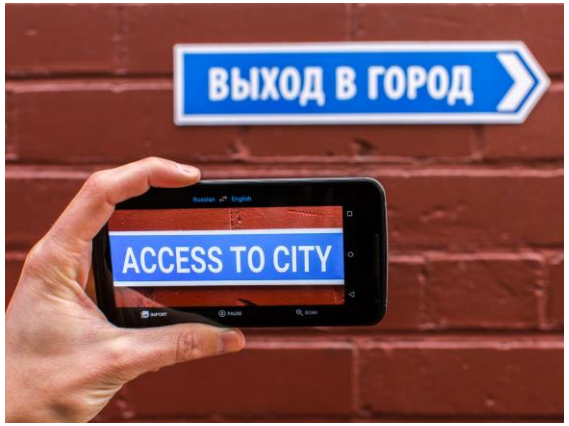
Autonomous driving:
Object detection



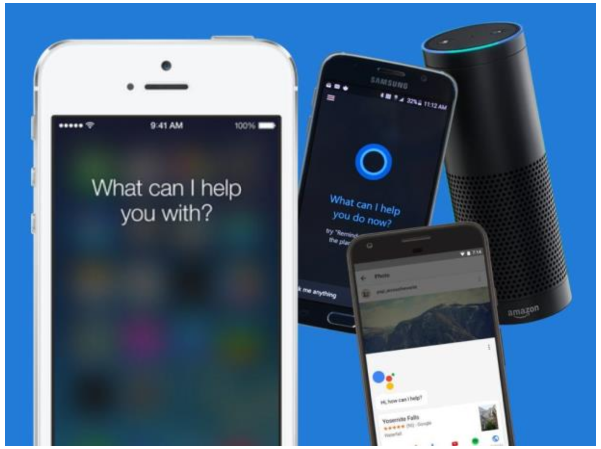
Medical imaging

Deep learning applications

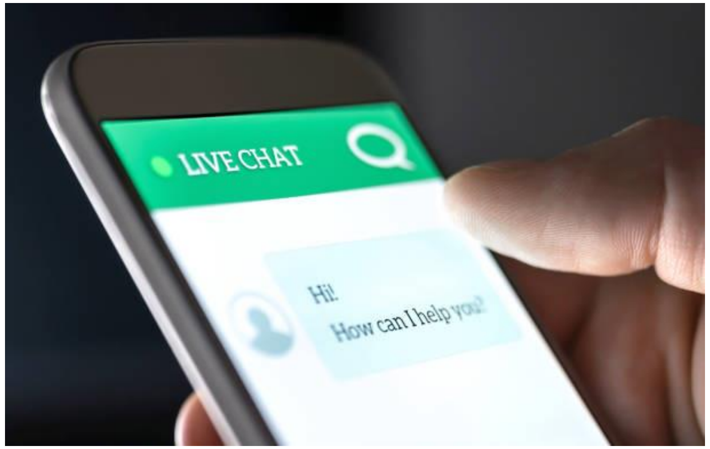
Natural language



Machine translation

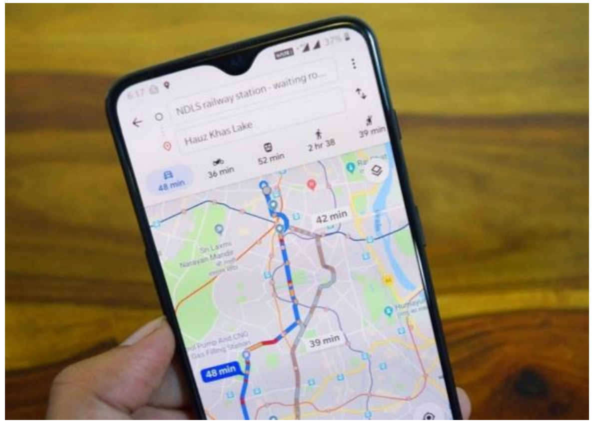


Virtual assistants



Chatbots

Deep learning applications

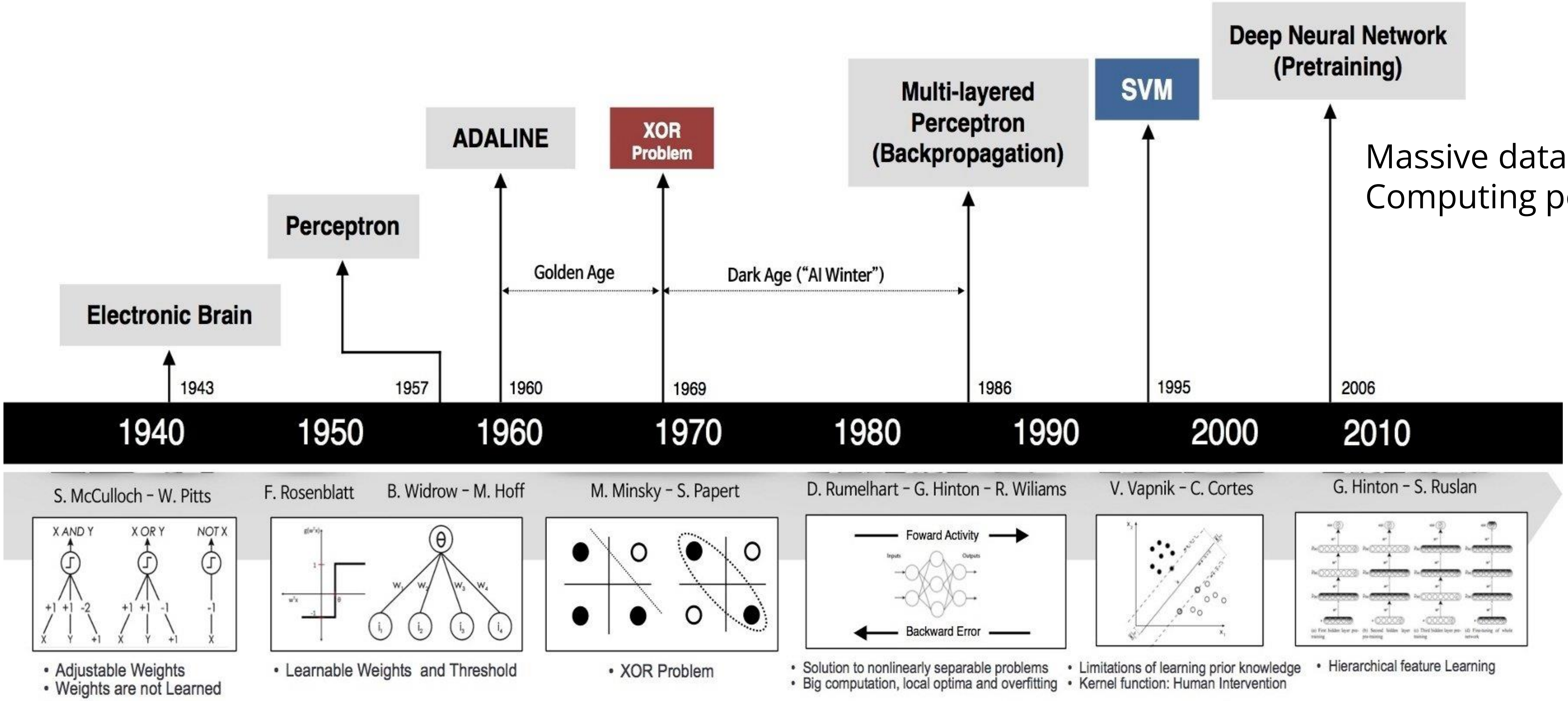


Navigation

Personalized
content

Image generation

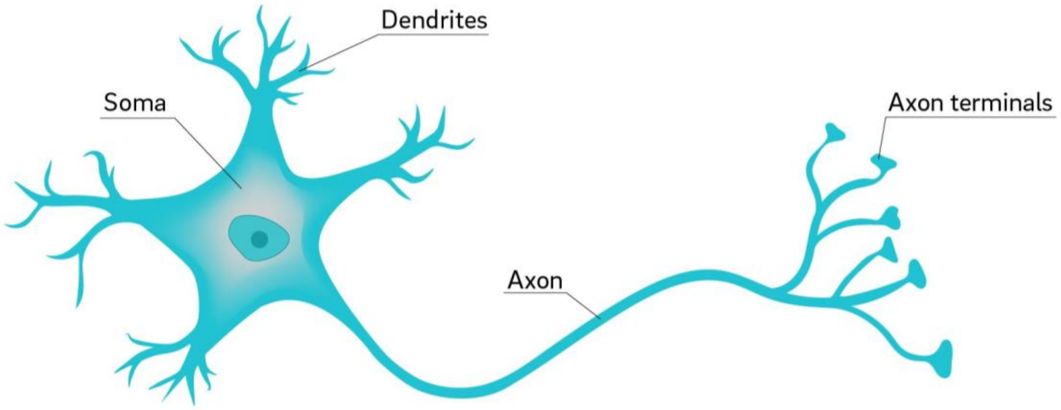
Neural network history



Massive data &
Computing power

Biological neural network

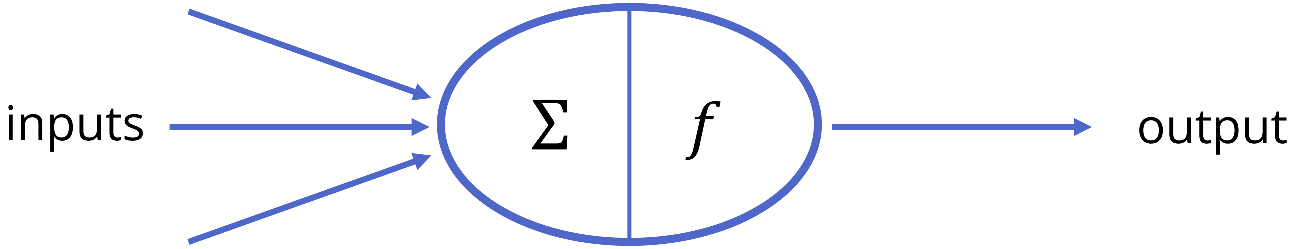
- ~86 billion neurons in the human brain
- On average, each neuron is connected to ~1000 others via synapses
- Electrical input "spikes"
- Fires if signal exceeds a threshold



[Image: David Baillot / UCSD]

Artificial neural network

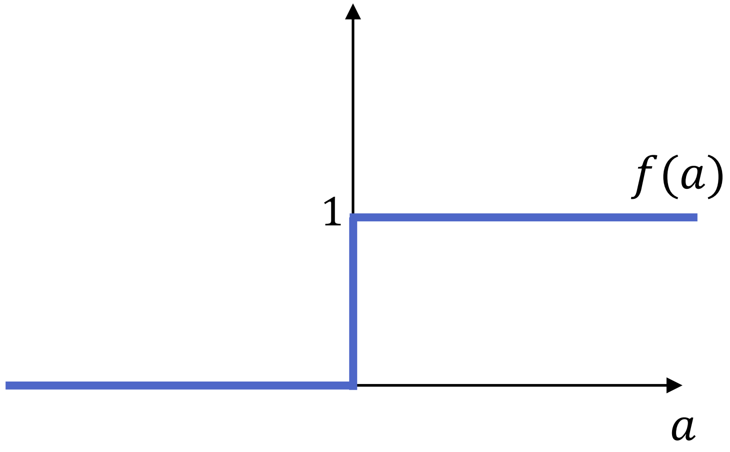
- Historical motivation: a simplified mathematical model of the biological neural network
- Get signals from previous neurons
- Aggregate them, and generate new signals
- Output signals to other neurons

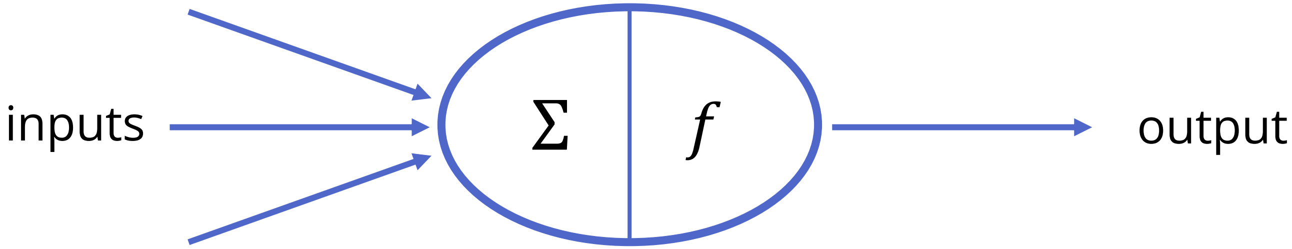


Perceptron

$$y(x_1, \dots, x_D) = f(w_1 x_1 + \cdots + w_D x_D + b)$$

$$f(a) = \begin{cases} 1 & \text{if } a \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

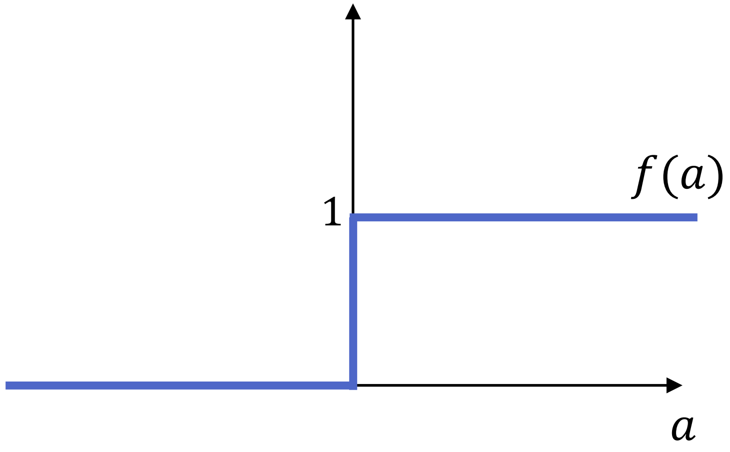


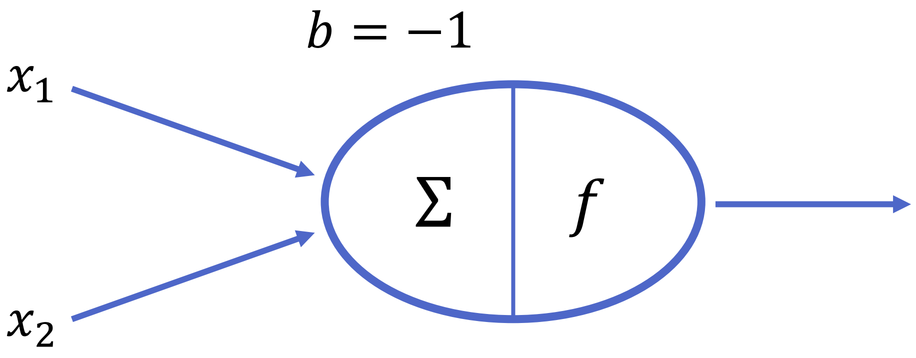


Perceptron

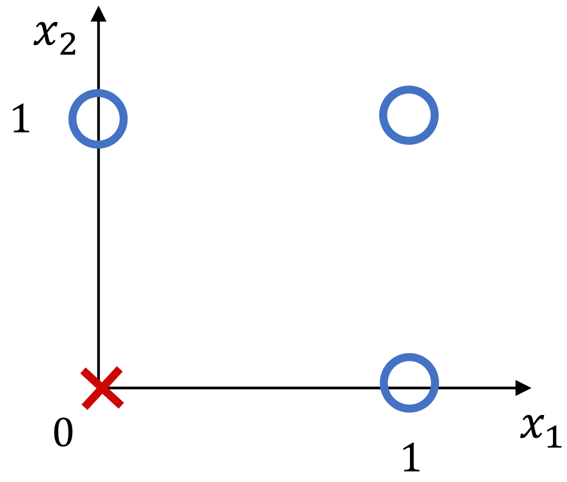
$$y(x_1, \dots, x_D) = f(w_1 x_1 + \cdots + w_D x_D + b)$$

$$f(a) = \begin{cases} 1 & \text{if } a \geq 0 \\ 0 & \text{otherwise} \end{cases}$$





$$\begin{aligned} &y(x_1, x_2) \\ &= f(x_1 + x_2 - 1) \\ &= x_1 \vee x_2 \end{aligned}$$

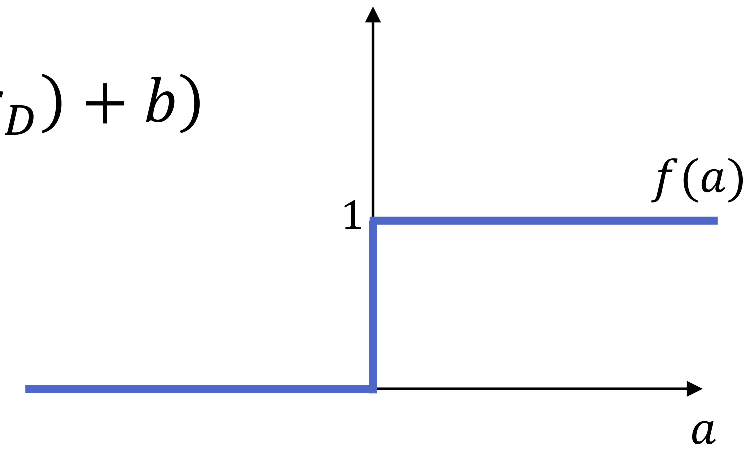


Multilayer perceptron (MLP)

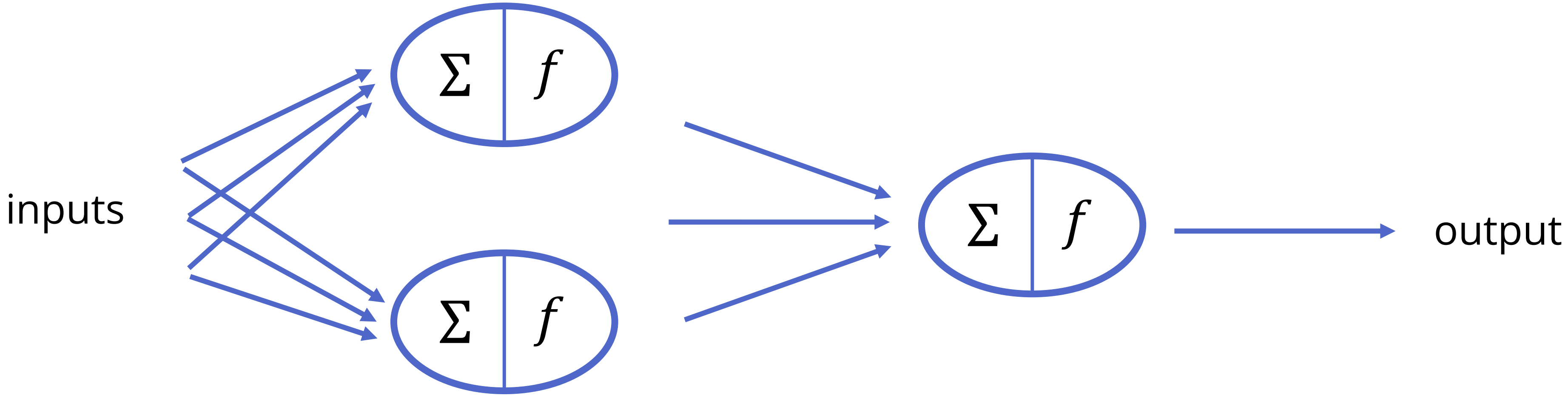
$$y(x_1, \dots, x_D) = f(w_1\phi_1(x_1, \dots, x_D) + \cdots + w_M\phi_M(x_1, \dots, x_D) + b)$$

$$f(a) = \begin{cases} 1 & \text{if } a \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Not differentiable!



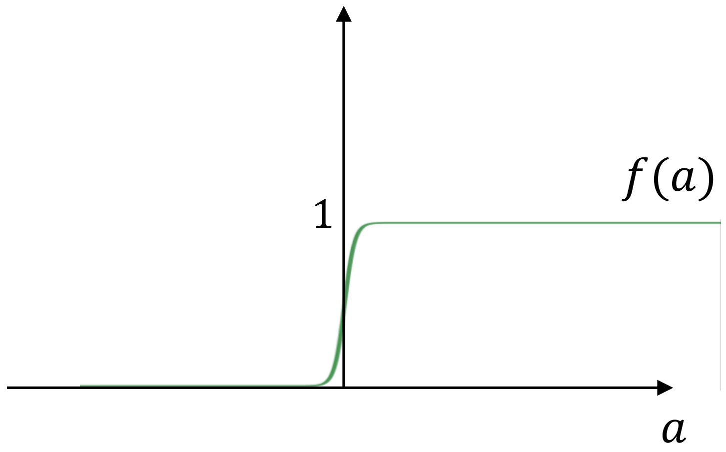
Suppose $\phi_i(x_1, \dots, x_N)$ also take the form “linear + activation”

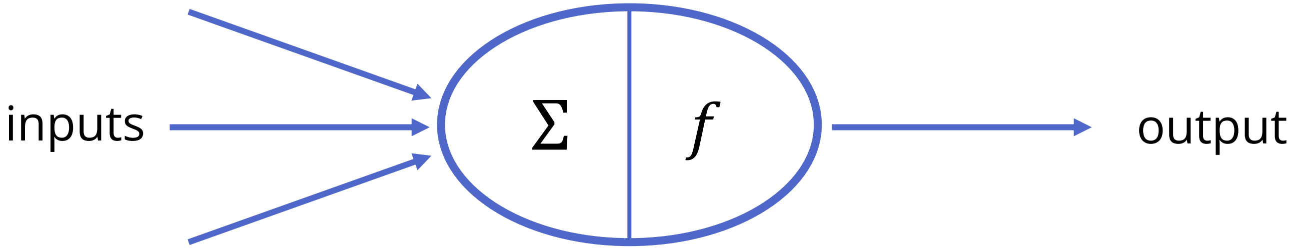


Logistic regression as a neuron

$$y(x_1, \dots, x_D) = f(w_1 x_1 + \cdots + w_D x_D + b)$$

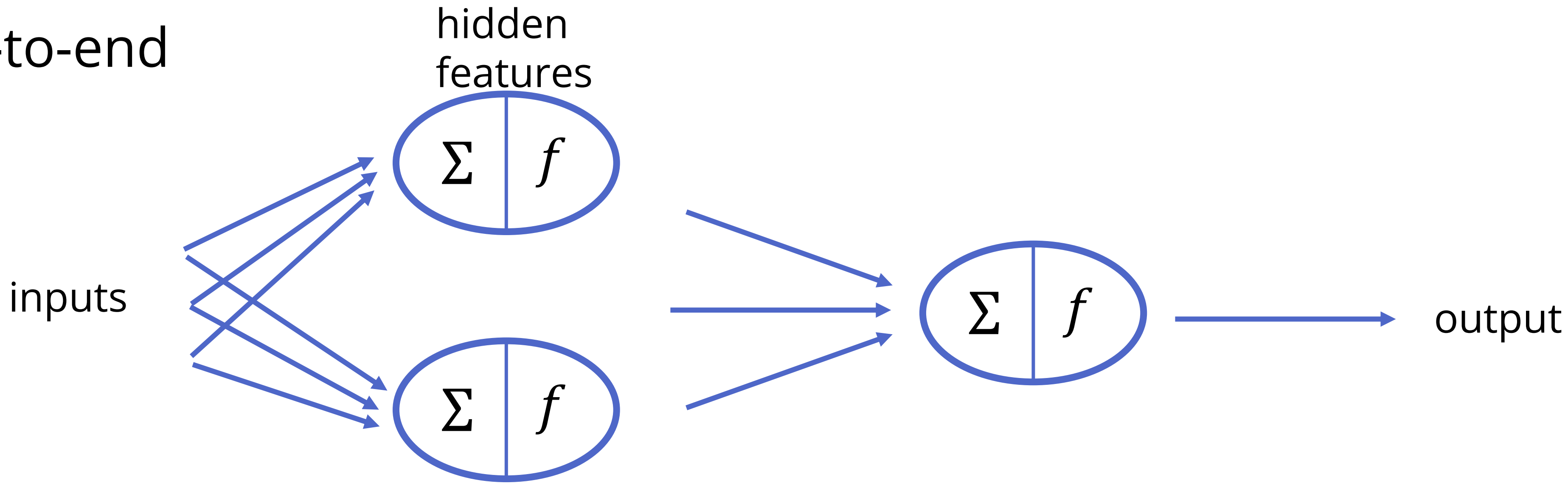
$$f(a) = \frac{1}{1 + \exp(-a)} = \sigma(a)$$





Multilayer perceptron

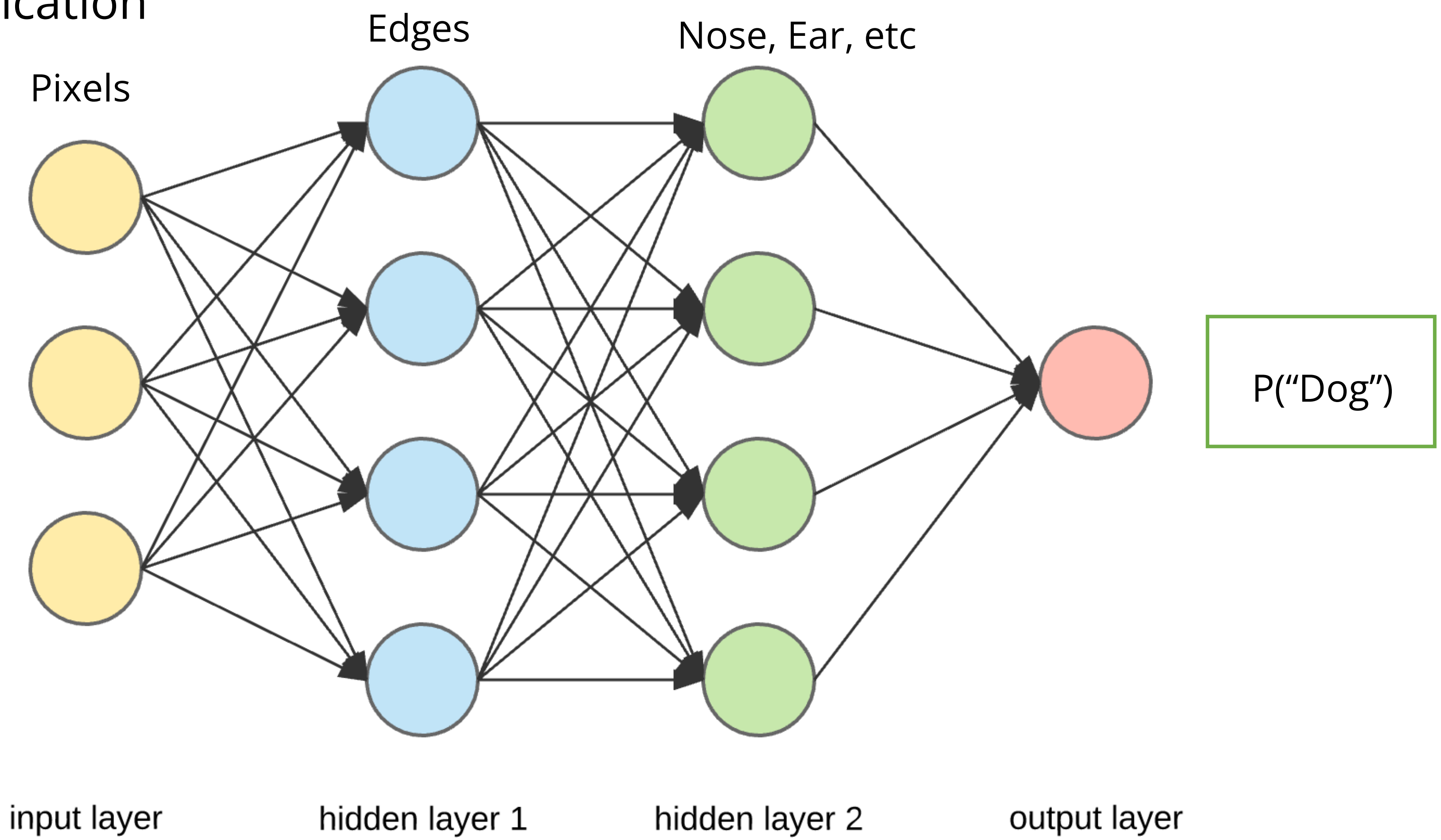
- *Feedforward neural network*
- Key idea: *learn* the features to fit the task (“adaptive basis functions”)
- Using sigmoid activation: effectively learn a “nested logistic regression” end-to-end



Multilayer perceptron

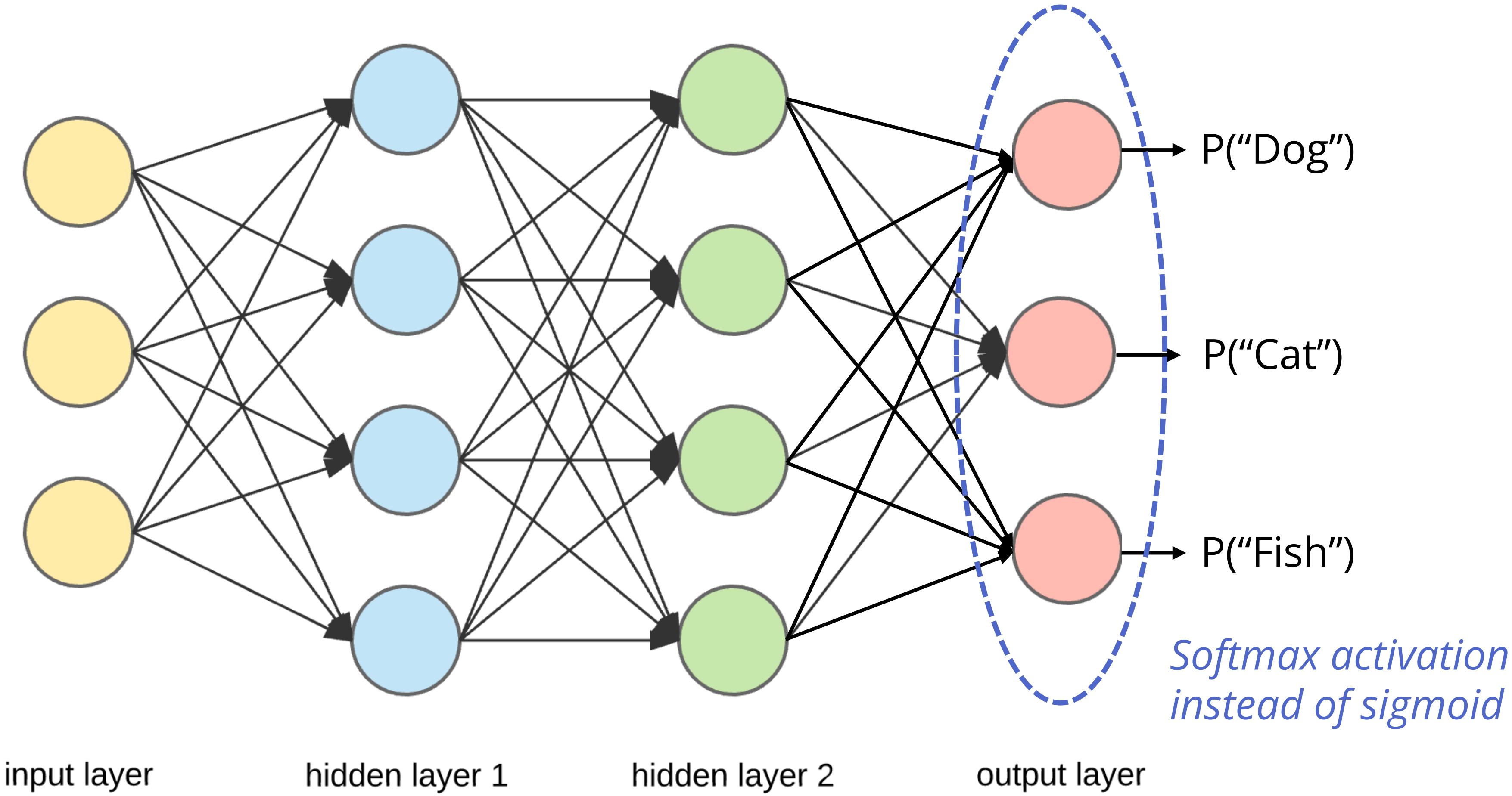
E.g. Image classification

Dog or not?

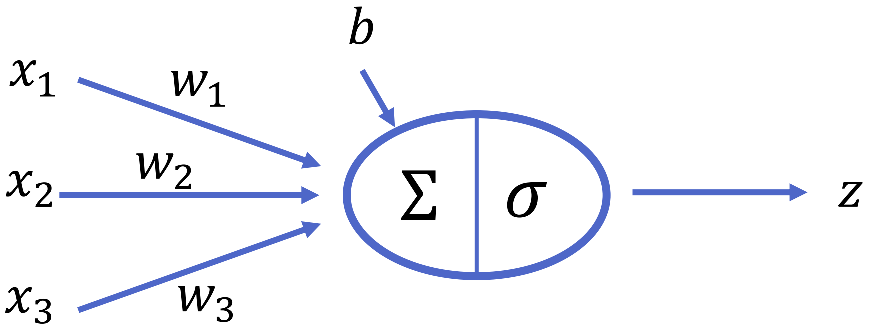


What about multiclass?

Dog, cat, or fish?

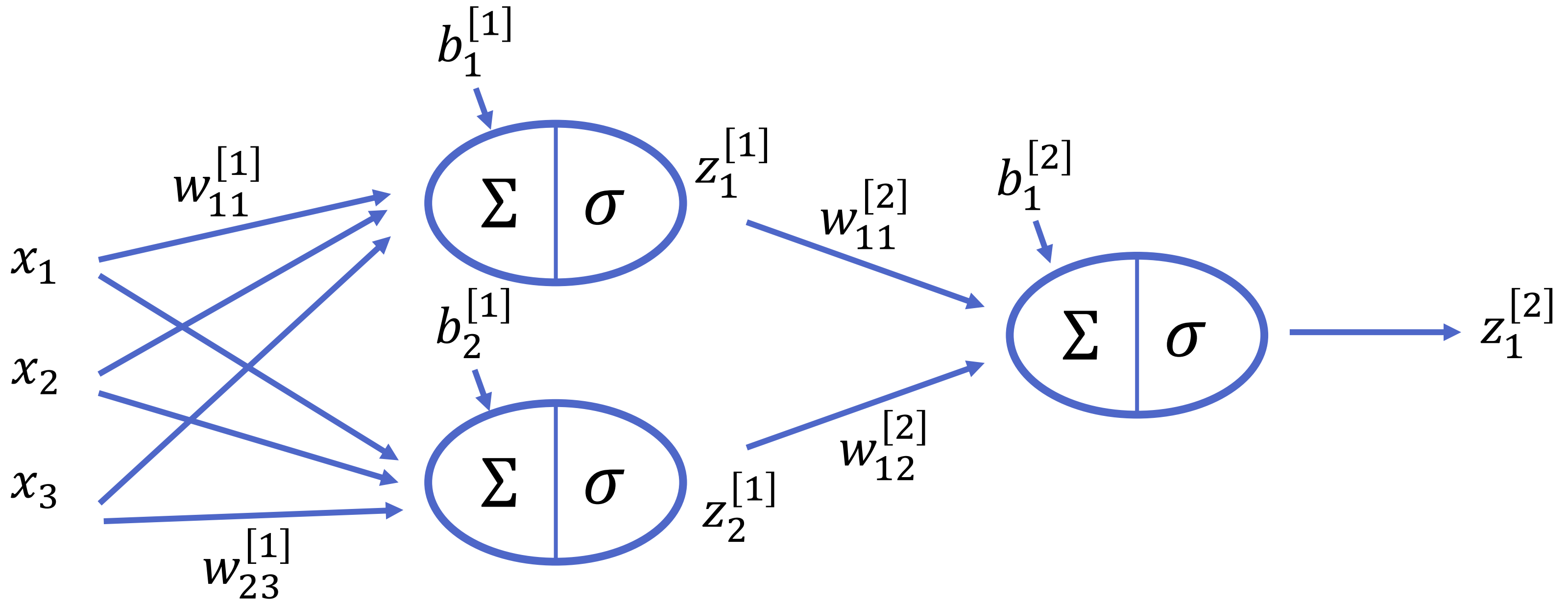


Forward propagation



$$a = \sum_{i=1}^{3} w_i x_i + b \qquad\qquad z = \sigma(a)$$

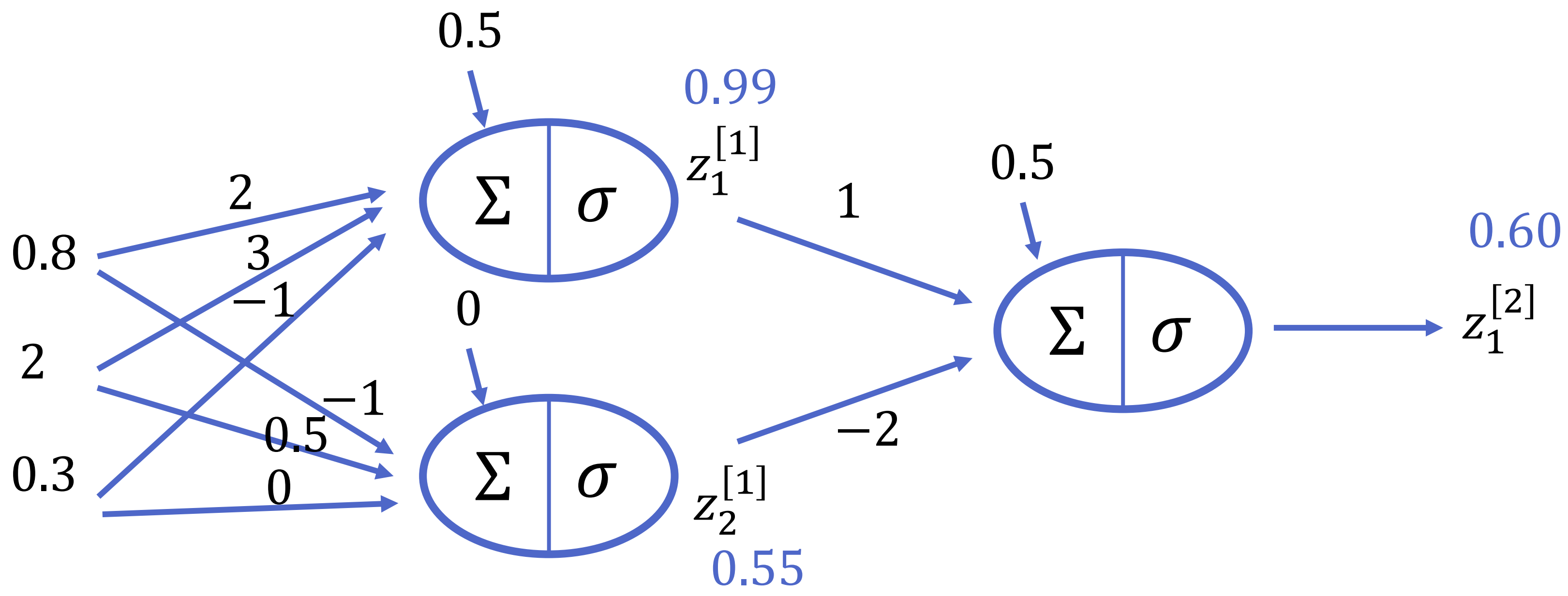
Forward propagation



$$a_1^{[1]} = \sum_{i=1}^{3} w_{1i}^{[1]} x_i + b_1^{[1]} \qquad z_1^{[1]} = \sigma\left(a_1^{[1]}\right) \qquad a_1^{[2]} = \sum_{i=1}^{2} w_{1i}^{[2]} z_i^{[1]} + b_1^{[2]}$$

$$a_2^{[1]} = \sum_{i=1}^{3} w_{2i}^{[1]} x_i + b_2^{[1]} \qquad z_2^{[1]} = \sigma\left(a_2^{[1]}\right) \qquad z_1^{[2]} = \sigma\left(a_1^{[2]}\right)$$

Forward propagation



$$a_1^{[1]} = 2(0.8) + 3(2) - 1(0.3) + 0.5$$
$$= 7.8$$

$$z_1^{[1]} = \sigma(7.8)$$
$$= 0.99$$

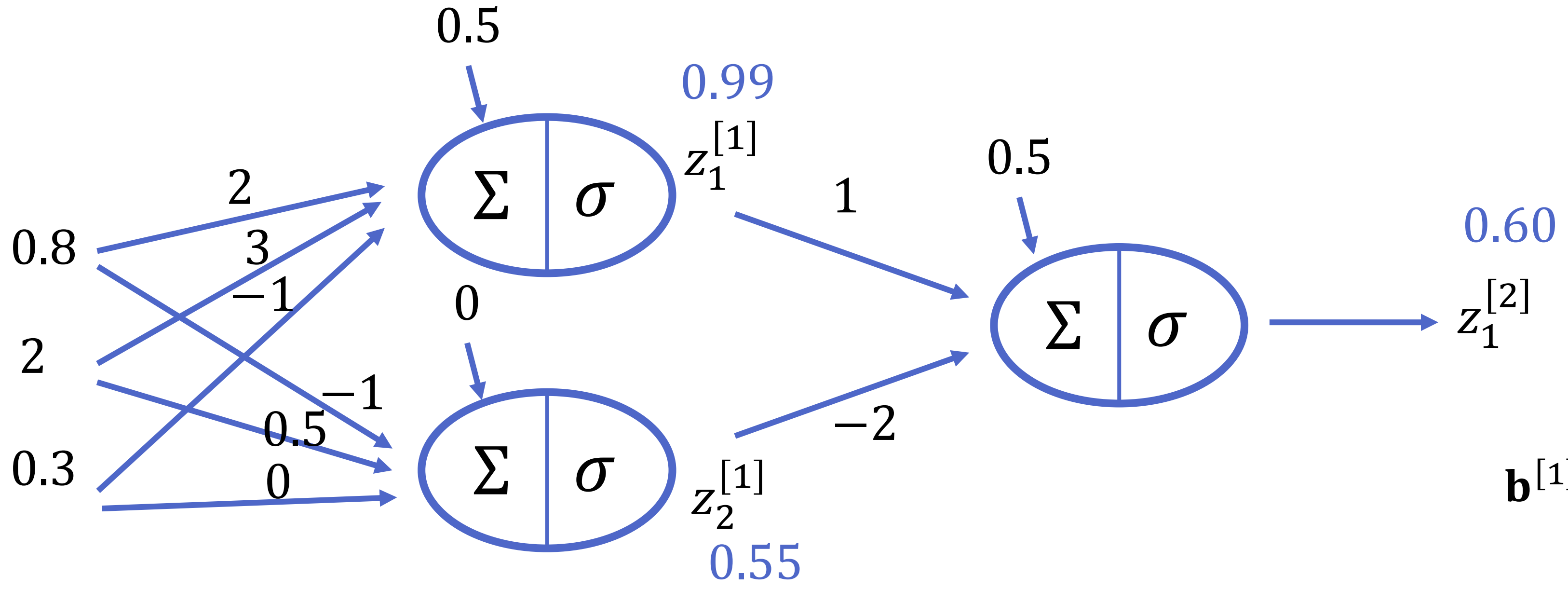
$$a_1^{[2]} = 1(0.99) - 2(0.55) + 0.5 = 0.39$$

$$a_2^{[1]} = -1(0.8) + 0.5(2) - 0(0.3) + 0$$
$$= 0.2$$

$$z_2^{[1]} = \sigma(0.2)$$
$$= 0.55$$

$$z_1^{[2]} = \sigma(0.39) = 0.60$$

Forward propagation



$$\mathbf{b}^{[1]} = \begin{bmatrix} 0.5 \\ 0 \end{bmatrix}$$

$$\mathbf{x} = \begin{bmatrix} 0.8 \\ 2 \\ 0.3 \end{bmatrix}$$

Weight matrices:

$$\mathrm{W}^{[1]} = \begin{bmatrix} w_{11}^{[1]} & w_{12}^{[1]} & w_{13}^{[1]} \\ w_{21}^{[1]} & w_{22}^{[1]} & w_{33}^{[1]} \end{bmatrix} = \begin{bmatrix} 2 & 3 & -1 \\ -1 & 0.5 & 0 \end{bmatrix}$$

$$\mathrm{W}^{[2]} = \begin{bmatrix} w_{11}^{[2]} & w_{12}^{[2]} \end{bmatrix} = \begin{bmatrix} 1 & -2 \end{bmatrix}$$

$$a^{[1]} = \mathrm{W}^{[1]}\mathbf{x} + \mathbf{b}^{[1]}$$

$$z^{[1]} = \sigma\left(a^{[1]}\right)$$

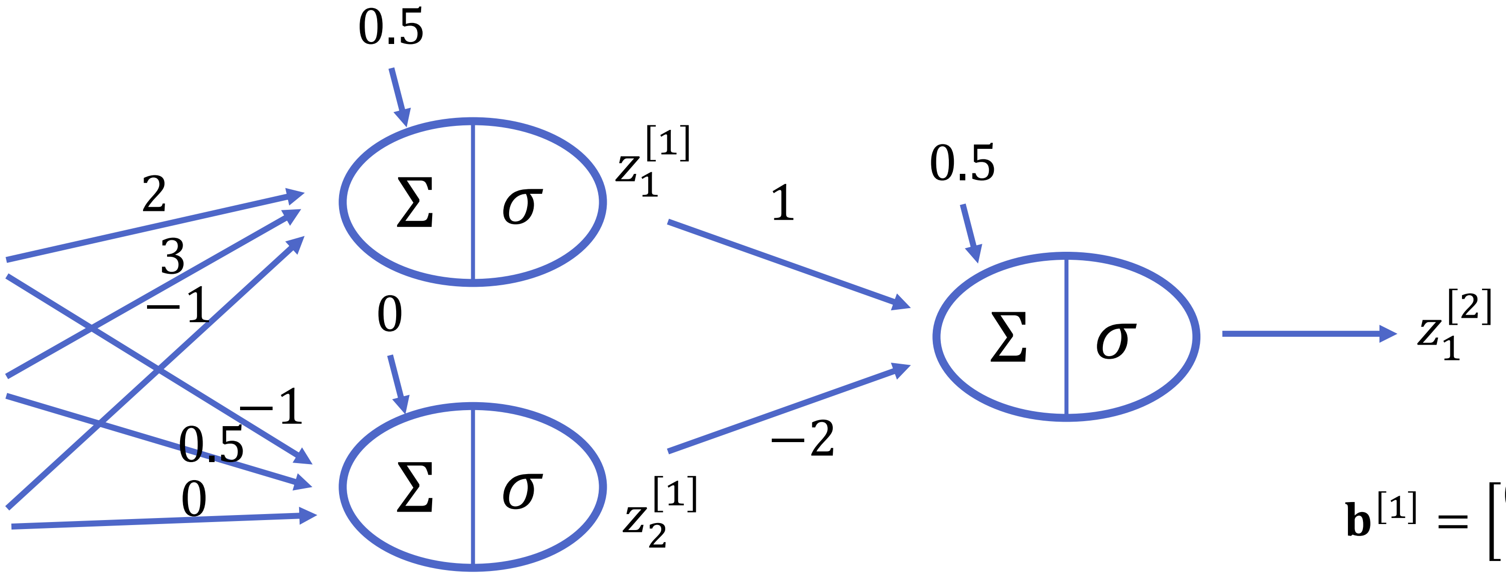
$$a^{[2]} = \mathrm{W}^{[2]}z^{[1]} + \mathbf{b}^{[2]}$$

$$z^{[2]} = \sigma\left(a^{[2]}\right)$$

Forward propagation: batch inputs

e.g. 4 examples

$$\begin{array}{cc} & \begin{array}{cccc} \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \mathbf{x}_4 \end{array} \\ \mathbf{X}^T = & \begin{bmatrix} 0.1 & 1 & -0.5 & 0.8 \\ 1 & -2 & 3 & 2 \\ 0.7 & 0.9 & 0.1 & 0.3 \end{bmatrix} \end{array}$$



broadcasting

$$\mathbf{b}^{[1]} = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Weight matrices:

$$\mathrm{W}^{[1]} = \begin{bmatrix} w_{11}^{[1]} & w_{12}^{[1]} & w_{13}^{[1]} \\ w_{21}^{[1]} & w_{22}^{[1]} & w_{33}^{[1]} \end{bmatrix} = \begin{bmatrix} 2 & 3 & -1 \\ -1 & 0.5 & 0 \end{bmatrix}$$

$$\mathrm{W}^{[2]} = \begin{bmatrix} w_{11}^{[2]} & w_{12}^{[2]} \end{bmatrix} = \begin{bmatrix} 1 & -2 \end{bmatrix}$$

$$a^{[1]} = \mathrm{W}^{[1]}\mathbf{X}^T + \mathbf{b}^{[1]}$$

$$\mathbf{b}^{[1]} = \begin{bmatrix} 0.5 \\ 0 \end{bmatrix}$$

$$z^{[1]} = \sigma\left(a^{[1]}\right)$$

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

$$z^{[2]} = \sigma\left(a^{[2]}\right)$$

Network training

- Parameters?
- Loss function? Binary classification: cross-entropy / logistic loss

$$E(\boldsymbol{W}) = -\sum_{n=1}^{N}\{t_n \ln y(\mathbf{x}_n) + (1 - t_n)\ln(1 - y(\mathbf{x}_n))\}$$

Forward propagation

- How to optimize? Gradient descent! (or variants)

Next up: backpropagation

d dimensional inputs
w units per hidden layer (width)

$W^{[1]}$: *w* x *d*
$W^{[2]}$: *w* x *w*
$W^{[3]}$: *1* x *w*