CS 232: Artificial Intelligence

Fall 2023

Prof. Carolyn Anderson
Wellesley College

Recap

Spot the differences

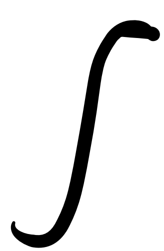
Neural Network Unit

$$z = b + \sum_i w_i x_i$$

$$y = \sigma(\mathbf{w} \cdot \mathbf{x} + b)$$

relu

tanh

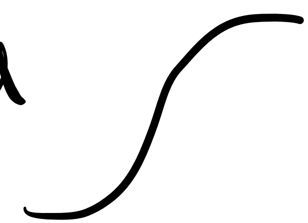


Logistic Regression

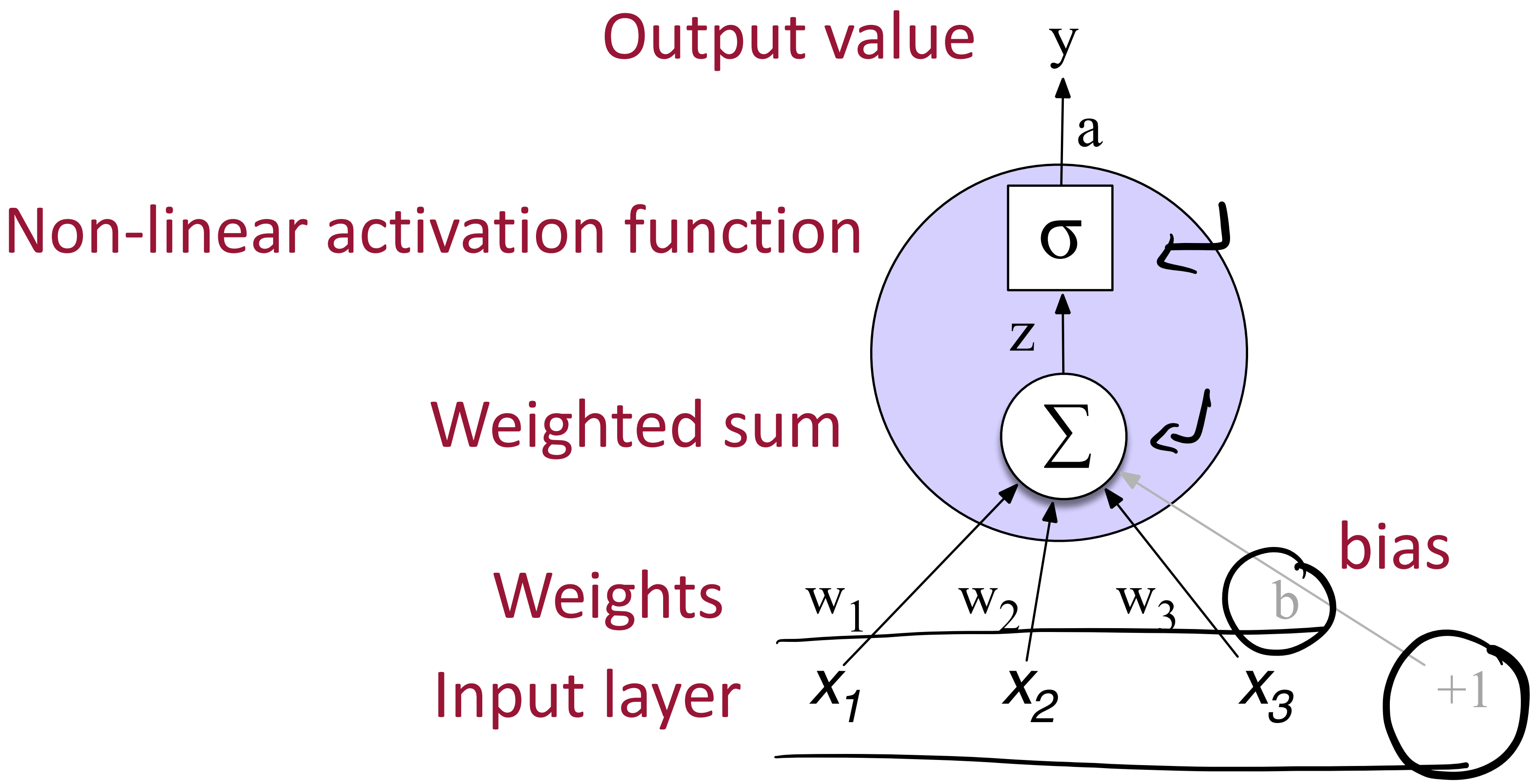
$$z = \left(\sum_{i=1}^{n} w_i x_i \right) + b$$

$$P(y = 1) = \sigma(w \cdot x + b)$$

Sigmoid



Final unit again



Example: XOR

The XOR problem

Minsky and Papert (1969)

Can neural units compute simple functions of input?

AND			OR			XOR		
x1	x2	y	x1	x2	y	x1	x2	y
0	0	0	0	0	0	0	0	0
0	1	0	0	1	1	0	1	1
1	0	0	1	0	1	1	0	1
1	1	1	1	1	1	1	1	0

Perceptrons

A very simple neural unit

- Binary output (0 or 1)
- No non-linear activation function

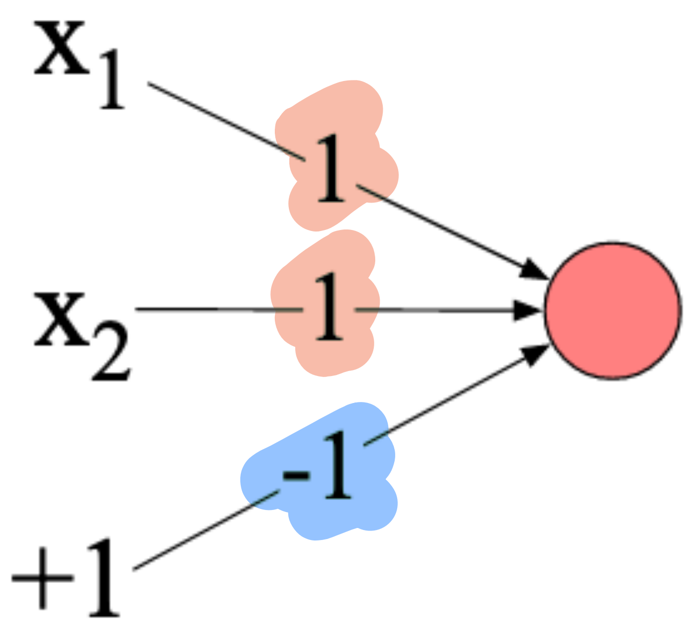
$$y = \begin{cases} 0, & \text{if } \mathbf{w} \cdot \mathbf{x} + b \leq 0 \\ 1, & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0 \end{cases}$$

Deriving AND

Goal: return 1 if x1 and x2 are 1

Use bias to make sure both must be 1

$$y = \begin{cases} 0, & \text{if } \mathbf{w} \cdot \mathbf{x} + b \leq 0 \\ 1, & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0 \end{cases}$$

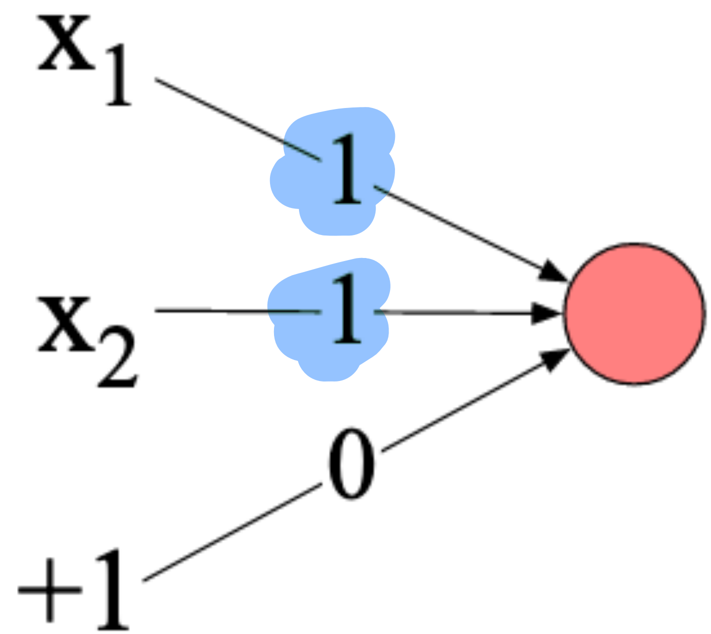


Deriving OR

Goal: return 1 if either input is 1

Don't need to do anything with bias

$$y = \begin{cases} 0, & \text{if } \mathbf{w}\cdot\mathbf{x}+b \leq 0 \\ 1, & \text{if } \mathbf{w}\cdot\mathbf{x}+b > 0 \end{cases}$$



solving XOR

$w_1 x_1 + w_2 x_2 + b \le 0$

if x_1 & x_2 are 1

$w_1 x_1 + w_2 x_2 + b \le 0$

if x_1 & x_2 are 0

$b \le 0$

$w_1 + w_2 + b \le 0$

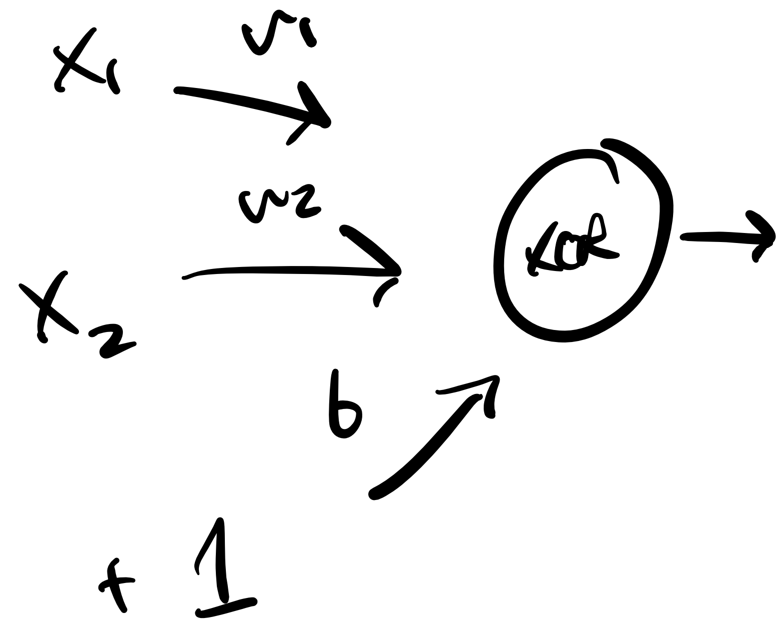
$w_1 x_1 + w_2 x_2 + b > 0$

if $x_1 = 1$ & $x_2 = 0$

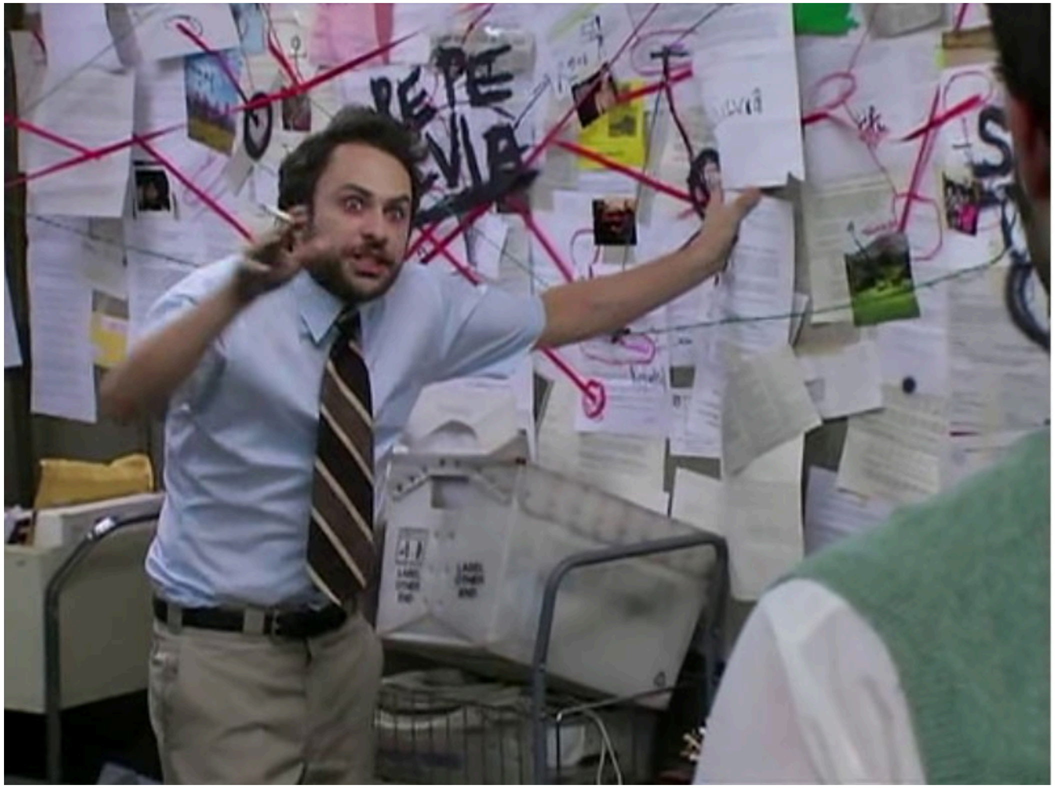
$w_1 + b > 0$

XOR

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0



Trick question!
It's not possible to capture XOR with perceptrons



Why? Perceptrons are linear classifiers

Perceptron equation is the equation of a line

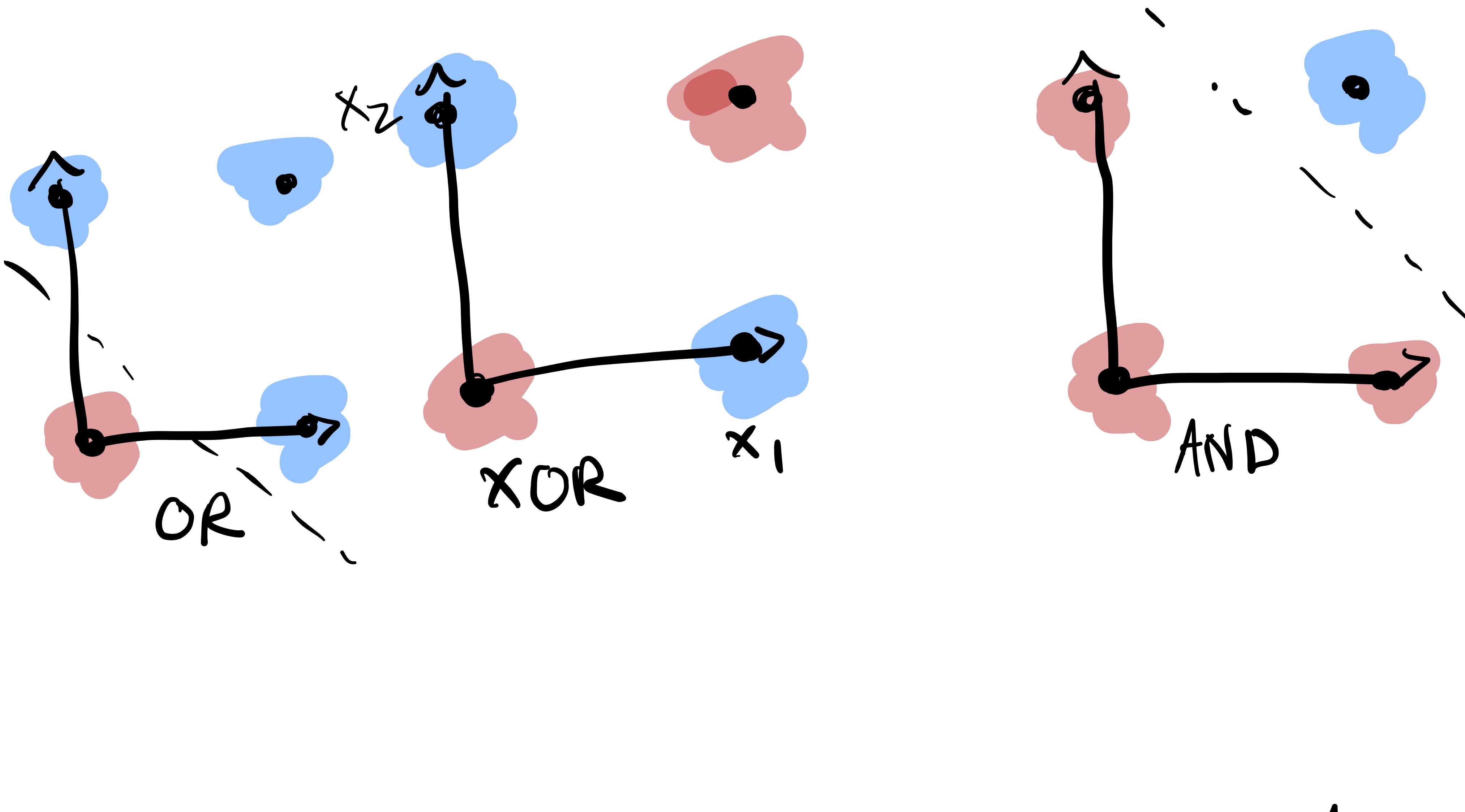
$$w_1 x_1 + w_2 x_2 + b = 0$$

(in standard linear format: $x_2 = (-w_1/w_2)x_1 + (-b/w_2)$)

This line acts as a **decision boundary**

- 0 if input is on one side of the line
- 1 if on the other side of the line

Decision boundaries



Solution to the XOR problem

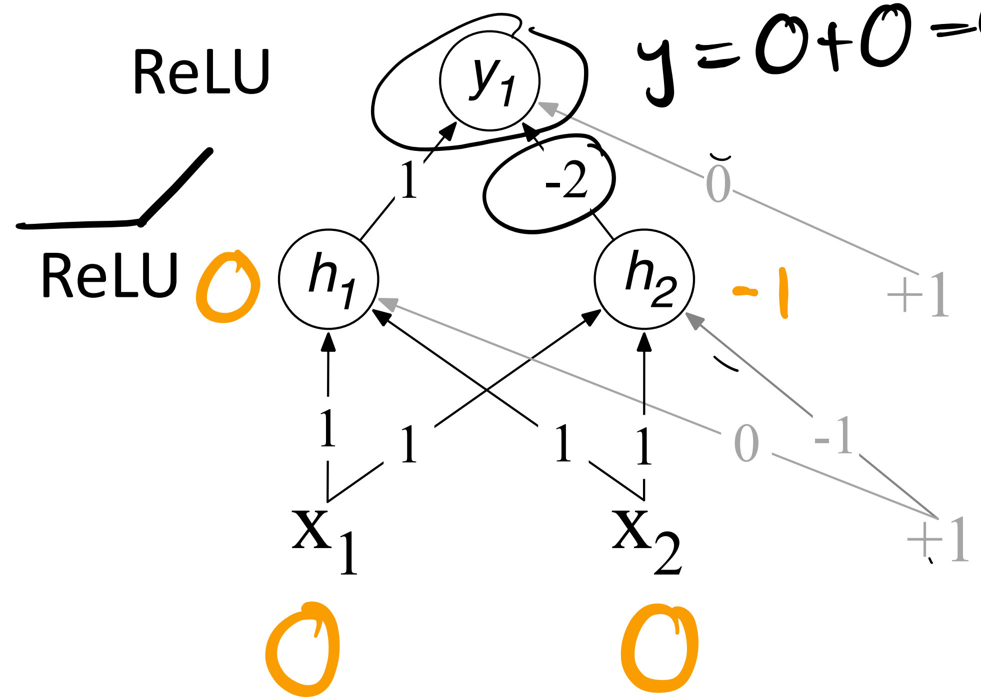
$y > 0 : 1$
$y \leq 0 : 0$
$\sigma(wx+b)$
$\uparrow$
h_1 & h_2

XOR **can't** be calculated by a **single** perceptron

XOR **can** be calculated by a layered network of units.

XOR

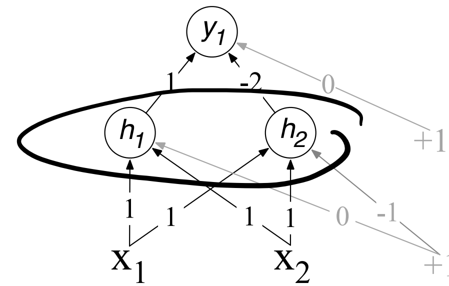
x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0

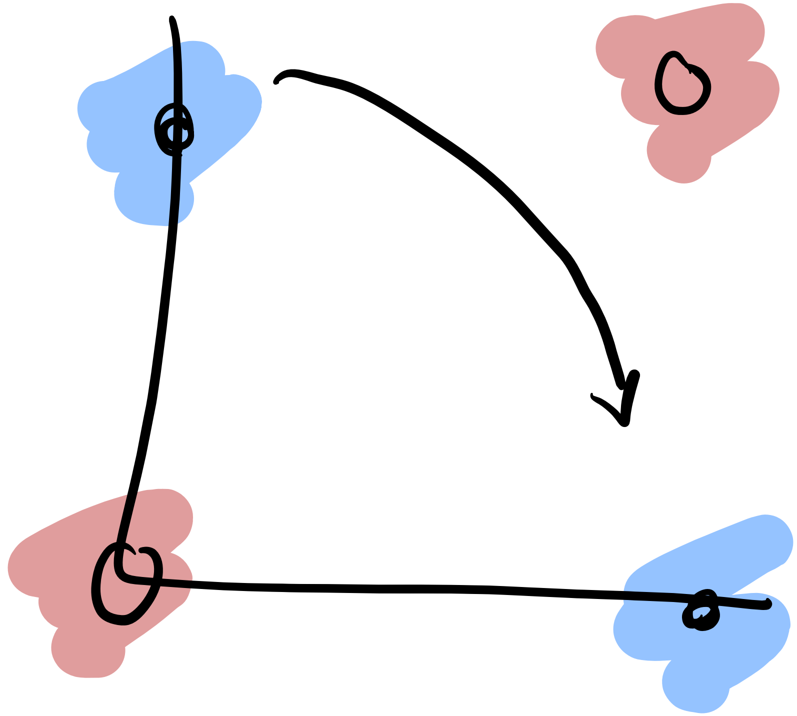


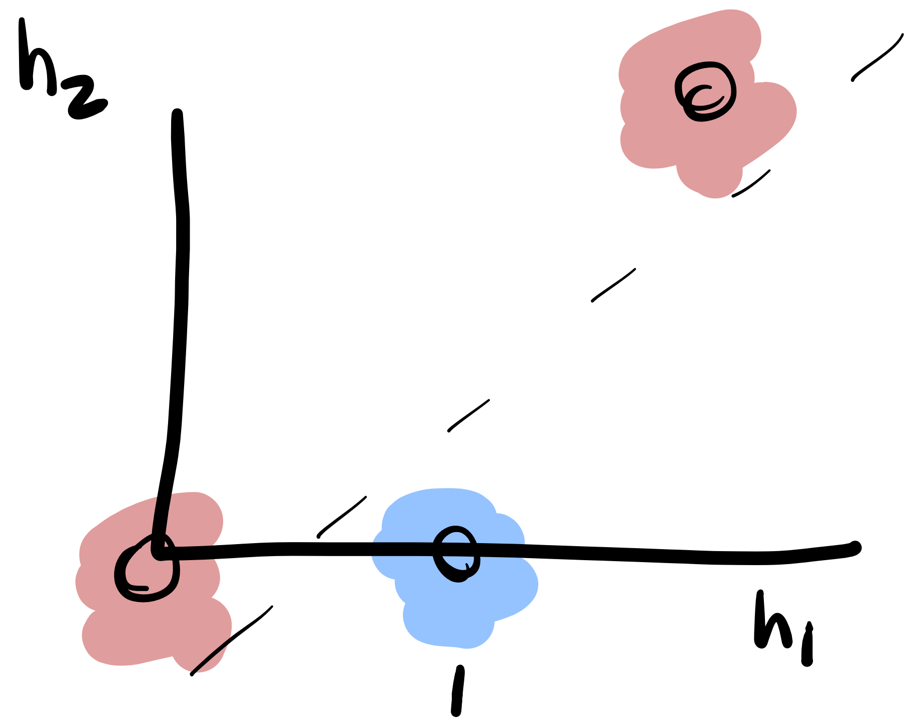
$y = 0 + 0 = 0$

reLu $f(x)$: x if $x > 0$
0 else

The hidden representation h





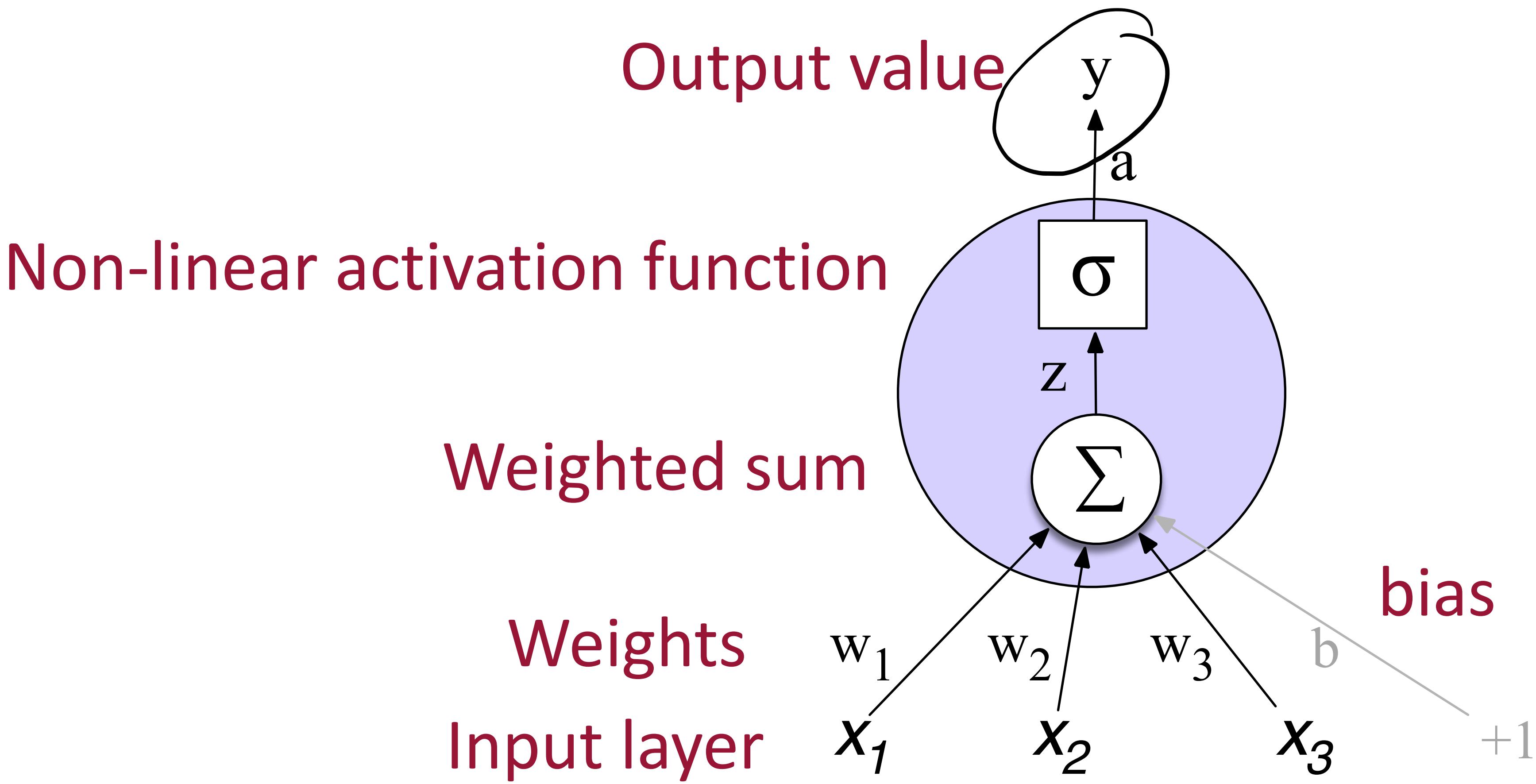


hidden layers: intermediate units

learn transformations of data

Feedforward Networks

Neural Network Unit



Multinomial Logistic Regression as a 1-layer Network

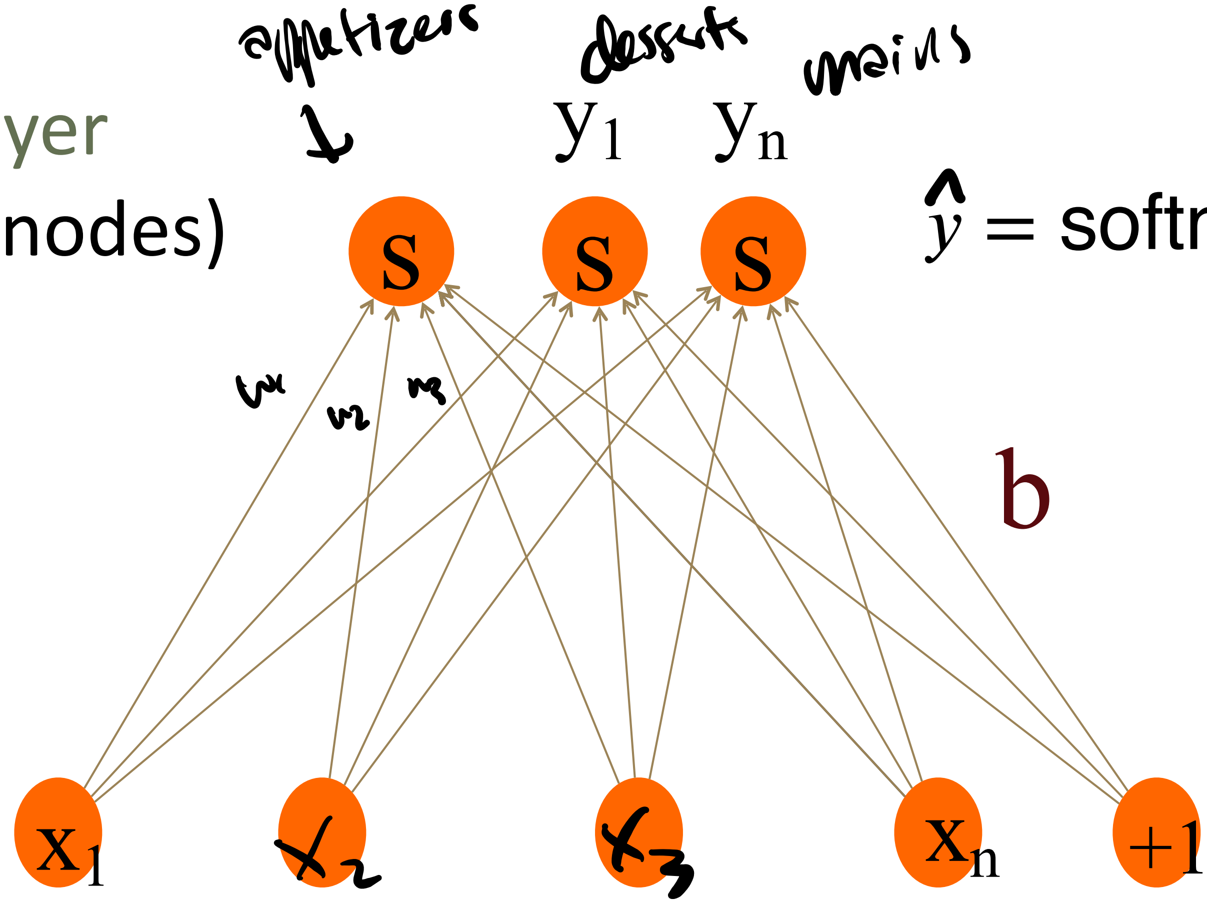
Fully connected single layer network

Output layer
(softmax nodes)

$$\hat{y} = \text{softmax}(Wx + b)$$

W

b



Input layer

Two-Layer Network with scalar output

Binary Classification

Output layer
(σ node)

$y = \sigma(z)$

$z = Uh$

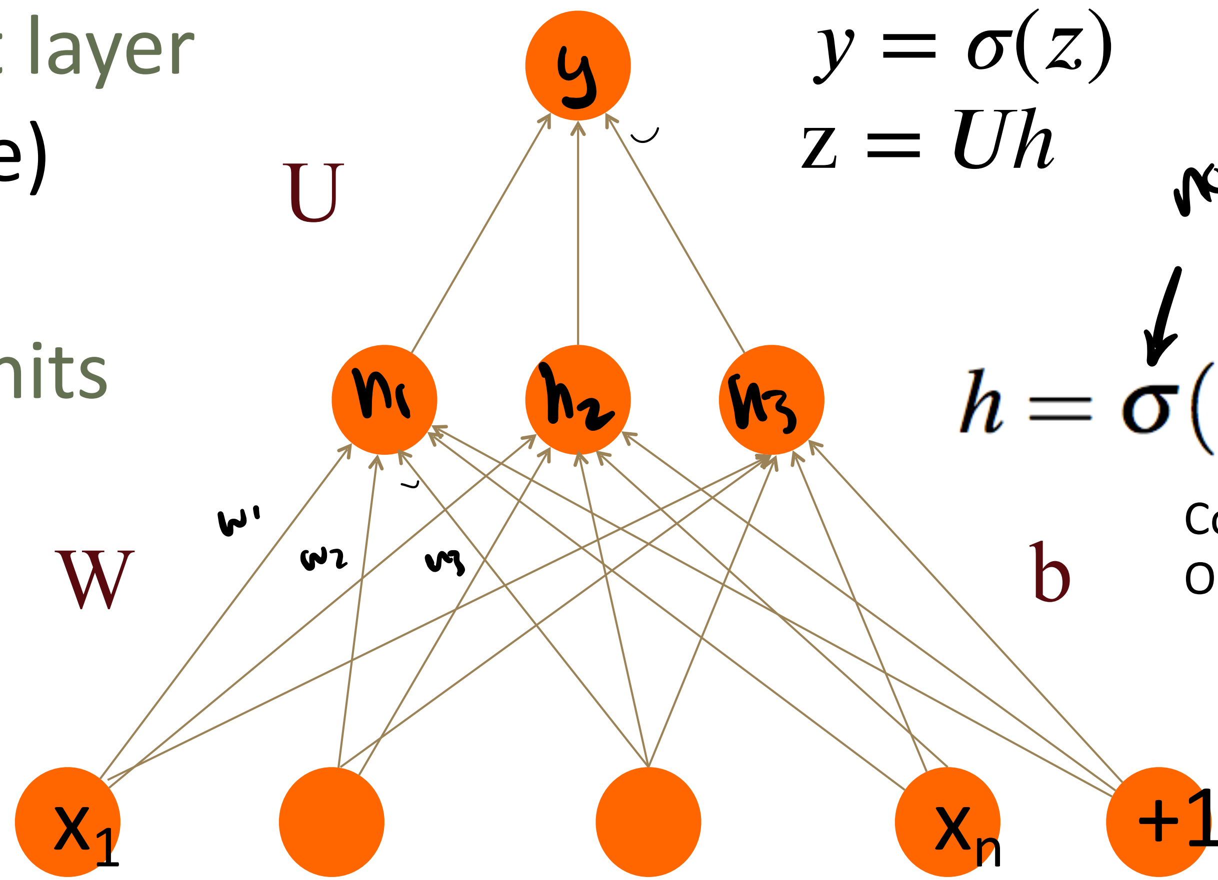
non-linear activation

hidden units
(σ node)

$h = \sigma(Wx + b)$

Could be ReLU
Or tanh

Input layer



bias

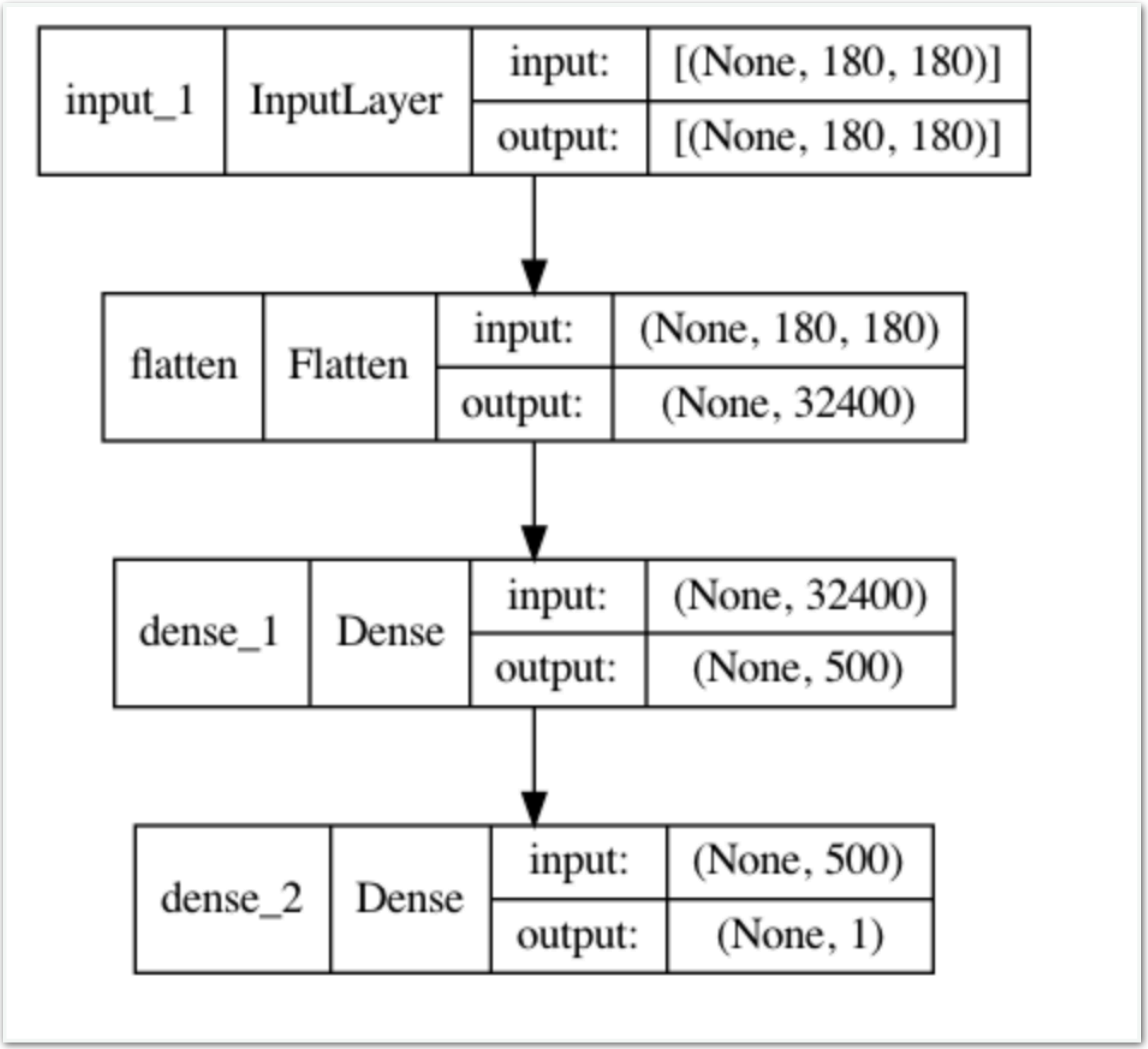
Using feedforward networks

Slides borrowed from Jurafsky & Martin Edition 3

Can we get back to cat pics, please?

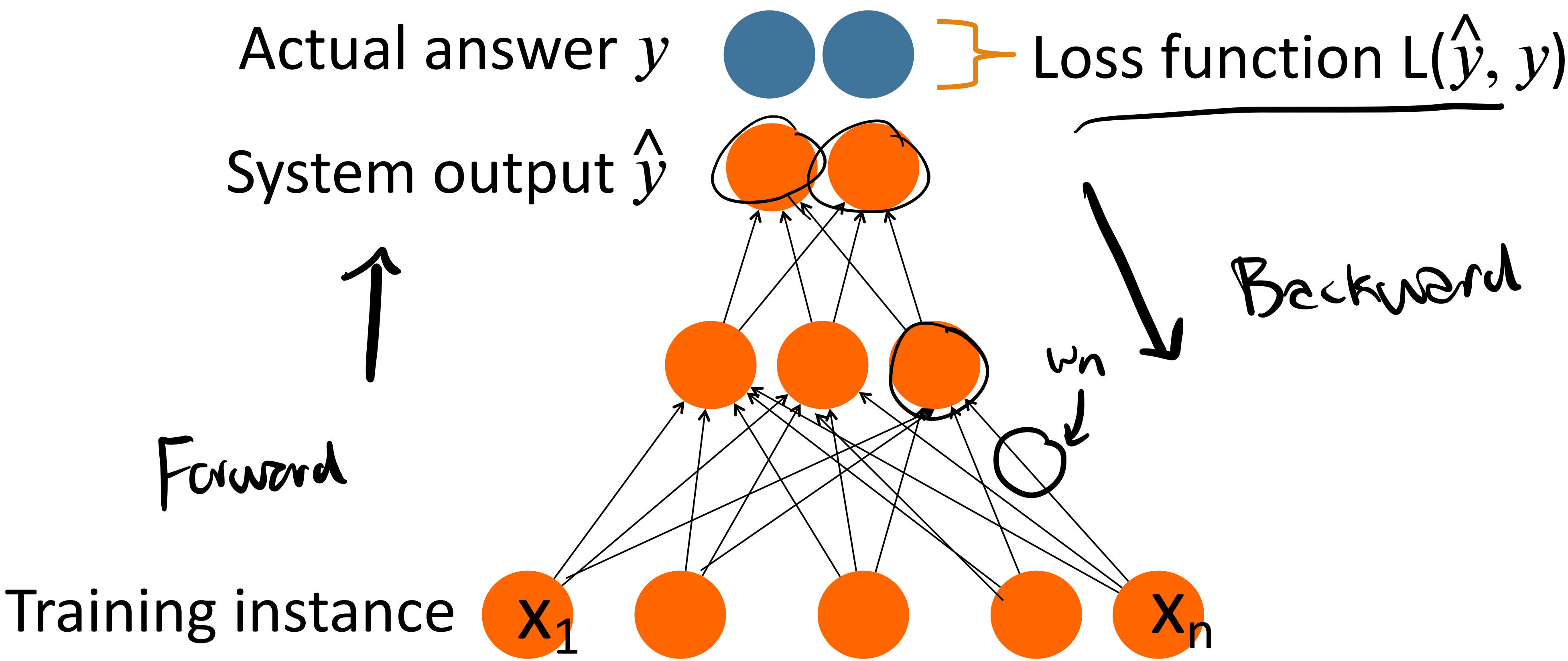
Finally, we're ready to power up our supervised cat/dog classifier by **adding more layers**. This takes it from a **regression model** to a **neural network**.

New Architecture



Training a Neural Network

Intuition: training a 2-layer Network



Intuition: Training a 2-layer network

For every training tuple (x, y)

- Run *forward* computation to find our estimate $\hat{y}$
- Run *backward* computation to update weights:
  - For every output node

    Compute loss L between y & $\hat{y}$

    For every weight w from hidden layer → output layer,
    update the weight

  - For every hidden node

    Assess how much blame it deserves

    For every weight w from input → hidden layer,
    update according to blame.