CS 232: AI

Spring 2024

Prof. Carolyn Anderson
Wellesley College

Reminders

- **Need to move and shorten my Friday help hours to 3-3:45**
- **Mid-semester updates with late days count coming soon**
- **Buy-one-get-three free policy on late days for this week**

YLLATAILY Discussion

Misinformation

Data gathering / data privacy

Too Good To Be True

Headlines that personify AI

Drawing on the last two chapters of YLLATAILY, come up with some rules of thumb for identifying misleading AI headlines

Trying to scare you

Responsibility shifting from humans to system

Task is too general

Task may have underlying biases

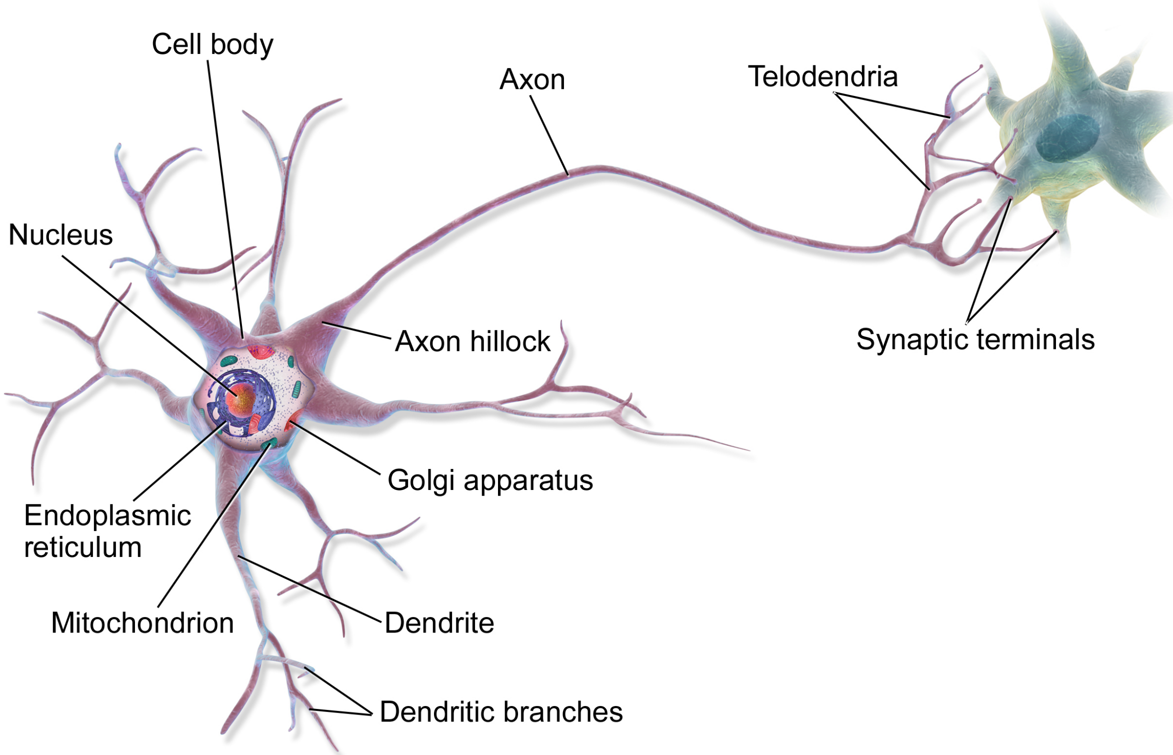
YLLATAILY Discussion

Neural Networks

This is a brain

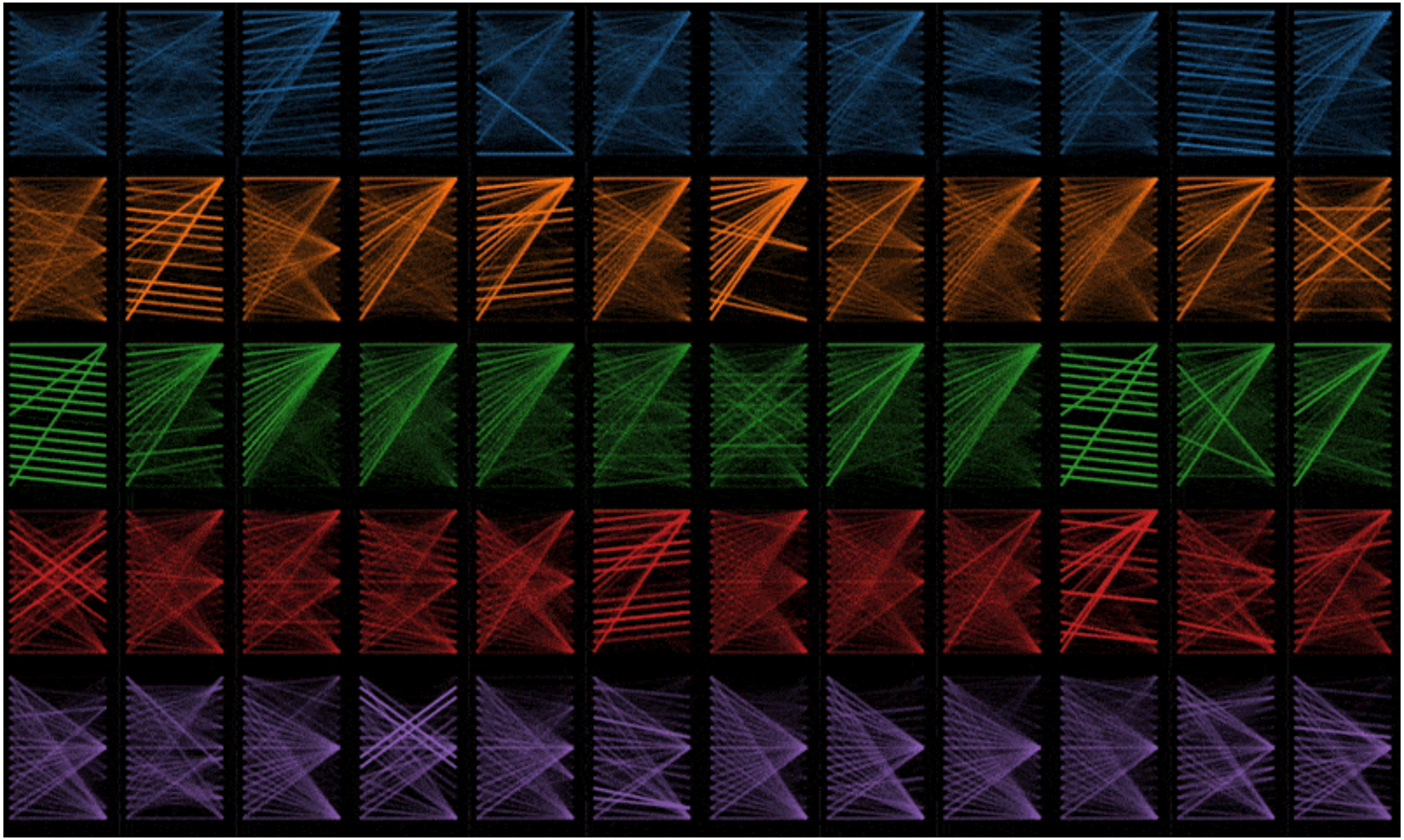
estimated to have 100 trillion synapses connecting 100 billion neurons

This is a brain



https://commons.wikimedia.org/w/index.php?curid=28761830

This is not your brain



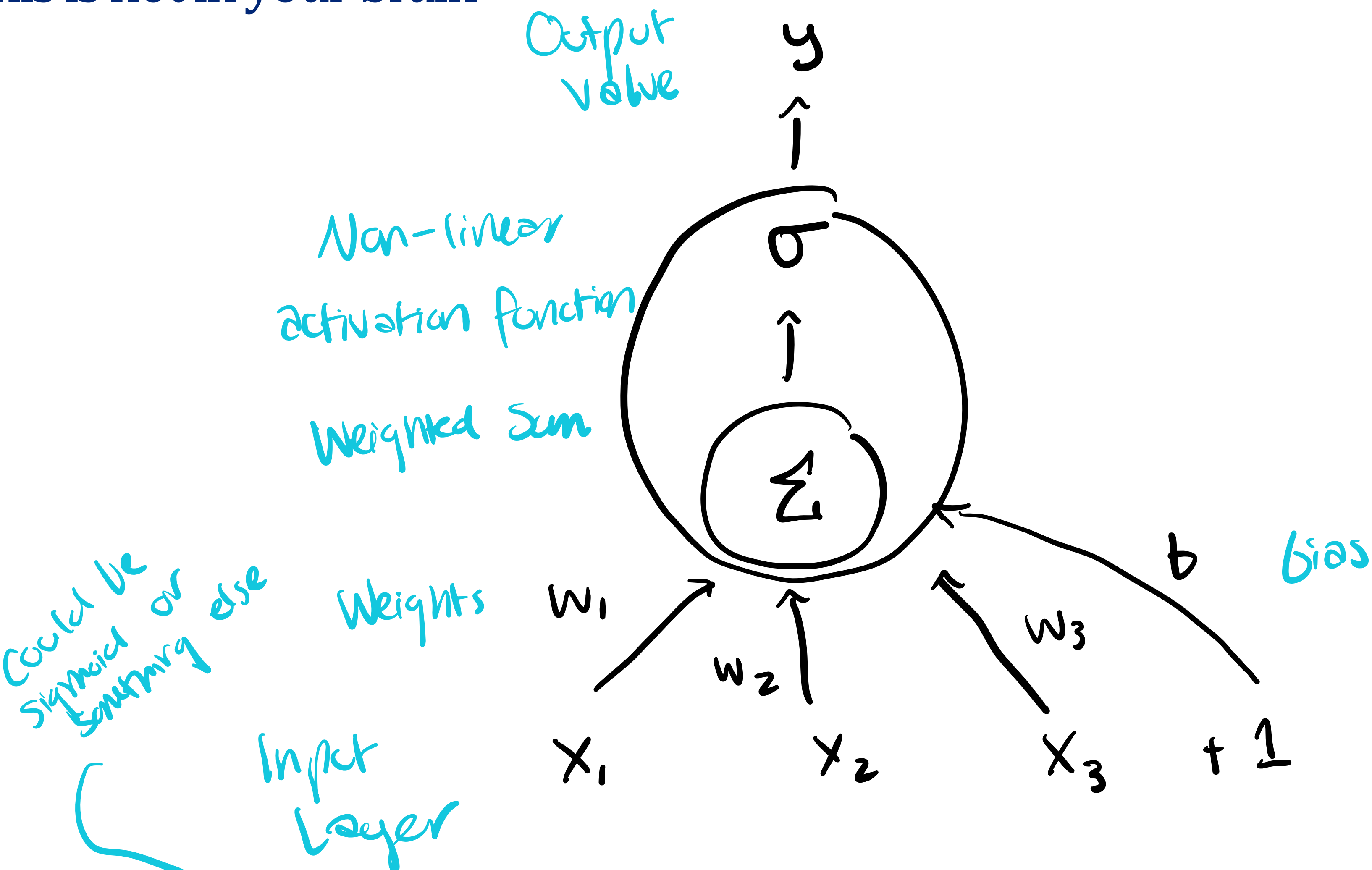
https://github.com/jessevig/bertviz

It's a large language model (neural network)

contemporary models have billions of parameters (GPT3: 175 billion)

Neural Network Unit

This is not in your brain



$$y = \sigma(z)$$

$$z = \sum wx + b$$

Units in Neural Networks

Neural unit

$$z = \left(\sum_i w_i x_i\right) + b$$

$$z = wx + b$$

$$y = f(z)$$

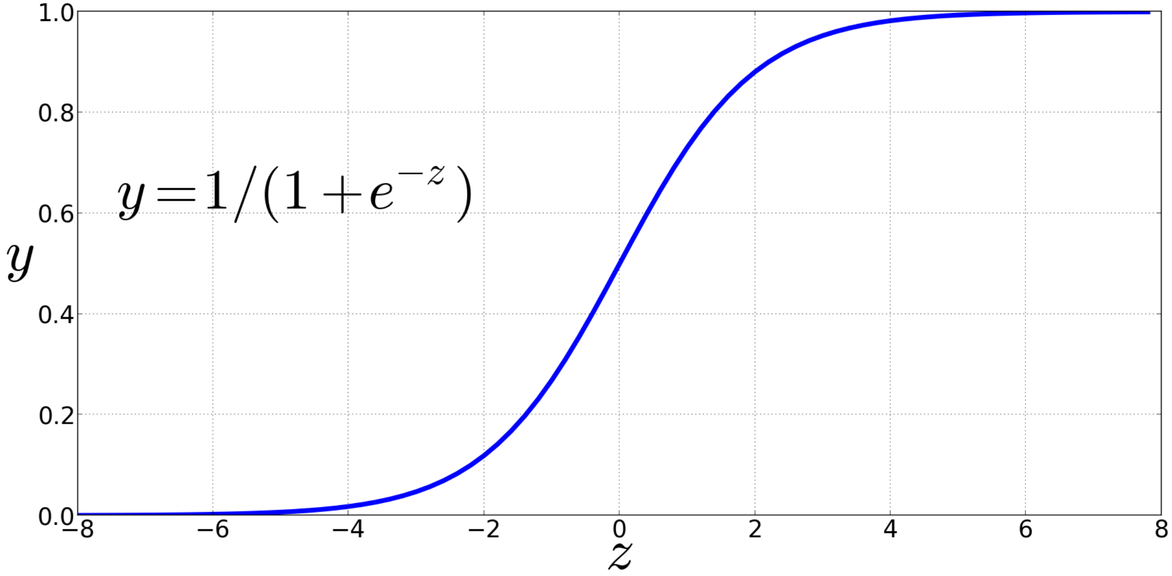
f could be σ

or something else

Non-Linear Activation Functions

We're already seen the sigmoid for logistic regression:

Sigmoid



Final function computed by a single unit

Spot the differences

Neural Network Unit

$$z = b + \sum_{i=1}^{n} w_i x_i$$

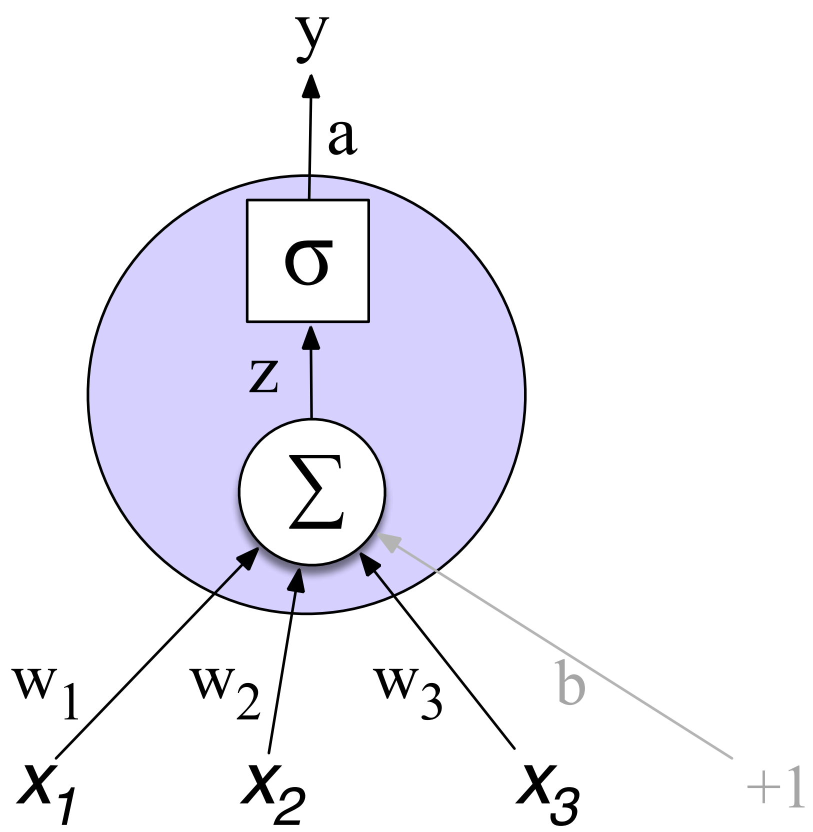
$$y = f(\overbrace{wx+b}^{z})$$

Logistic Regression

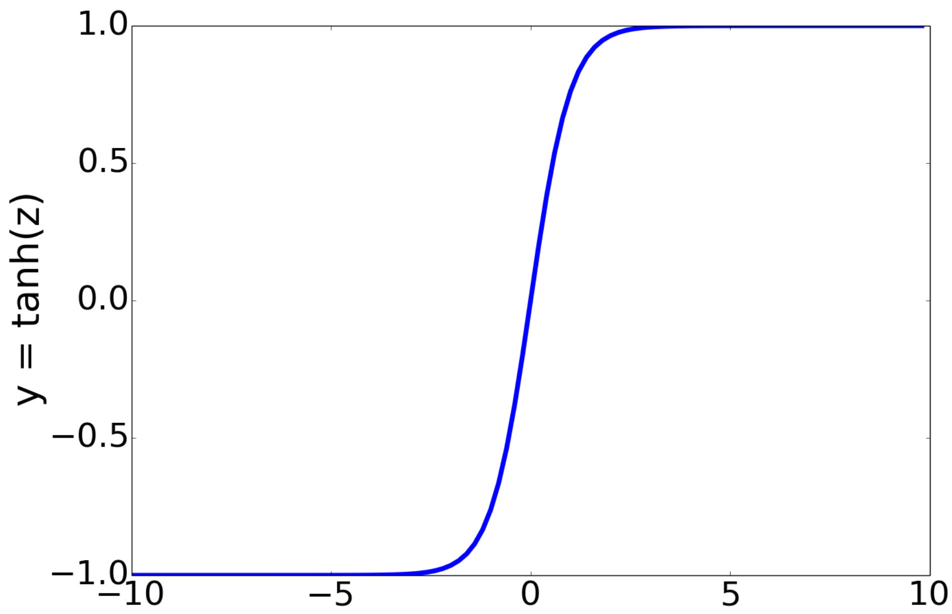
$$z = \left(\sum_{i=1}^{n} w_i x_i\right) + b$$

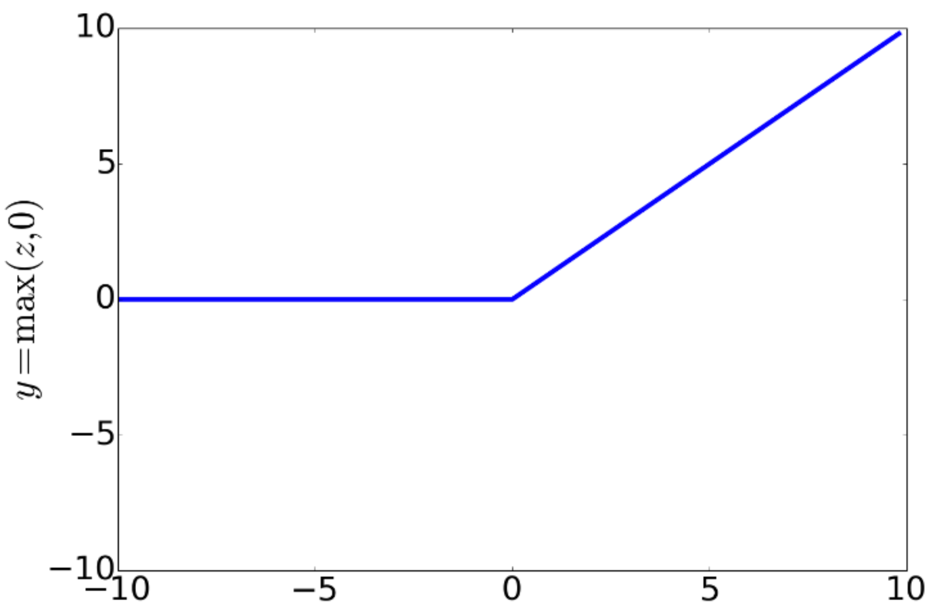
$$p(y=1 \mid x) = \sigma(\overbrace{wx+b}^{z})$$

Final unit again



Non-Linear Activation Functions besides sigmoid





tanh

$$y = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

ReLU

Rectified Linear Unit

$$y = max(z, 0)$$

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 unit

A very simple neural unit

- Binary output (0 or 1)
- A simple thresholding output function in place of sigmoid:

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

Solving AND

Deriving AND

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

AND

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

Goal: return 1 if x1 and x2 are 1

Deriving AND

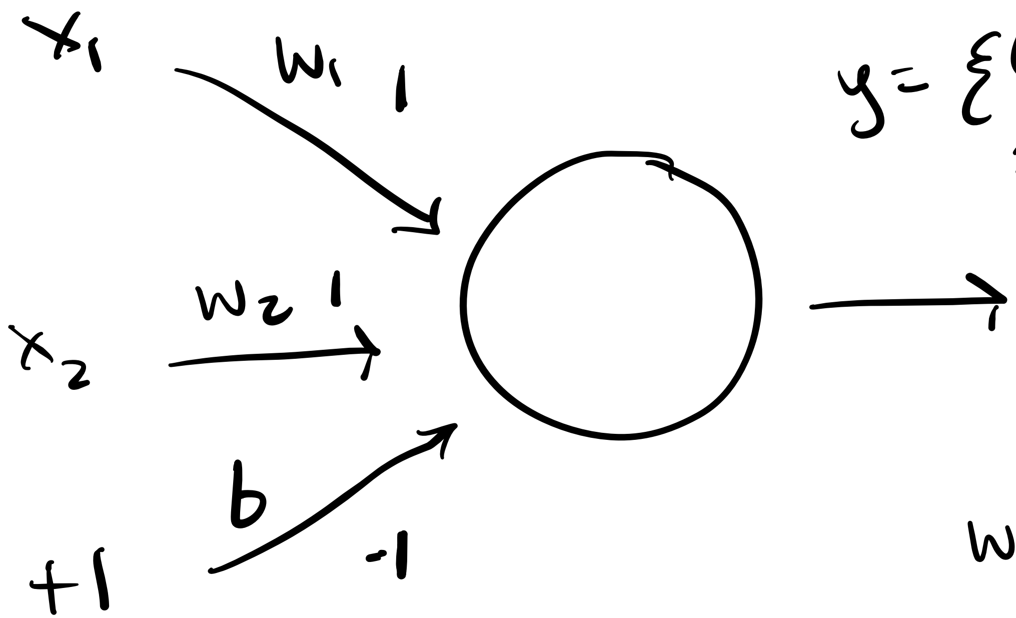
Goal: return 1 if x1 and x2 are 1

$w_1 + w_2 + b > 0$

$w_1 + b \le 0$

$w_2 + b \le 0$

$y = \begin{cases} 0 & \text{if } wx + b \le 0 \\ 1 & \text{if } wx + b > 0 \end{cases}$



$x_1 = 0$

$x_2 = 1$

Goal:

$y = 0$

$z = 0 + 1 - 1$

$= 0$

$y = 0$

$x_1 = 1$

$x_2 = 1$

Goal: $y = 1$

$z = 1 + 1 + -1$

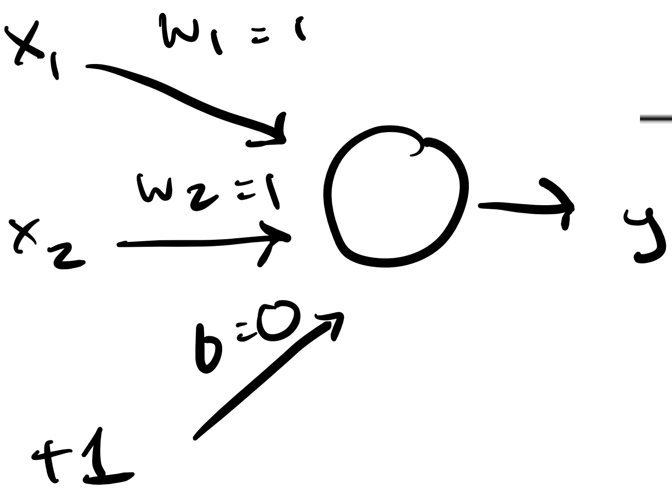
$= 1$

$y = 1$

$w_1 x_1 + w_2 x_2 + b > 0$ if x_1, x_2 are both 1

Exercise: solving OR

$$y = \begin{cases} 0 & \text{if } z \le 0 \\ 1 & \text{if } z > 0 \end{cases}$$



OR

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

Deriving OR

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

Goal: return 1 if either input is 1

OR

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

Deriving OR

Goal: return 1 if either input is 1

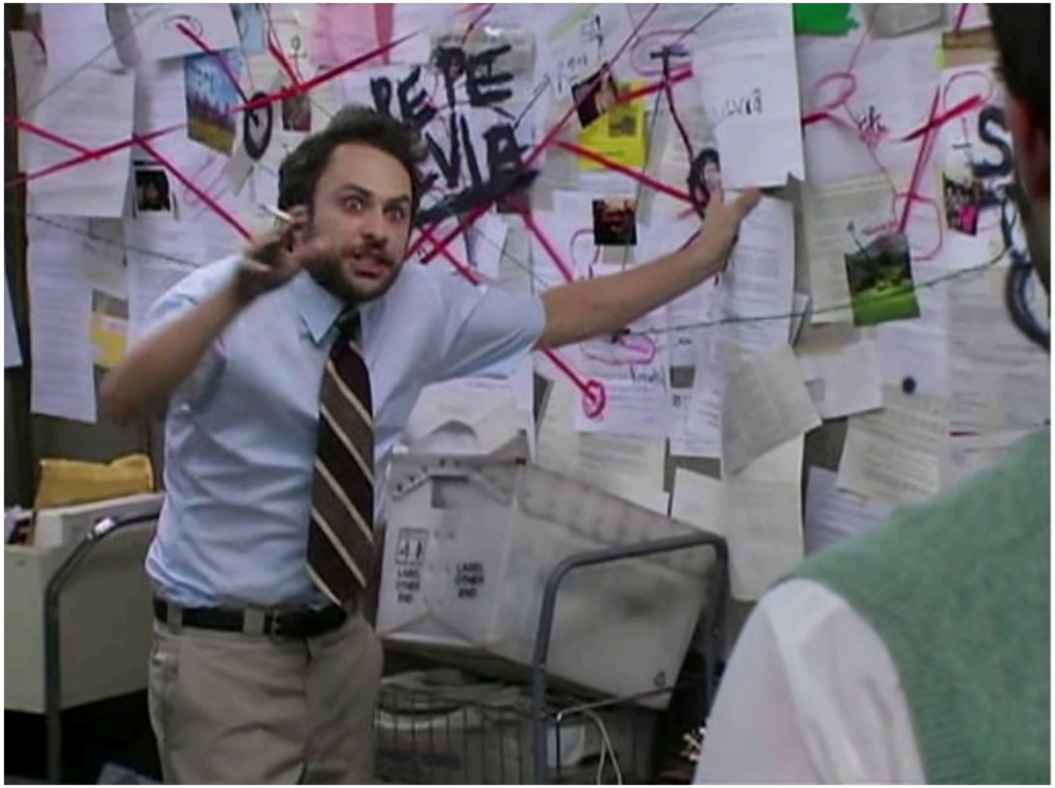
solving XOR

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

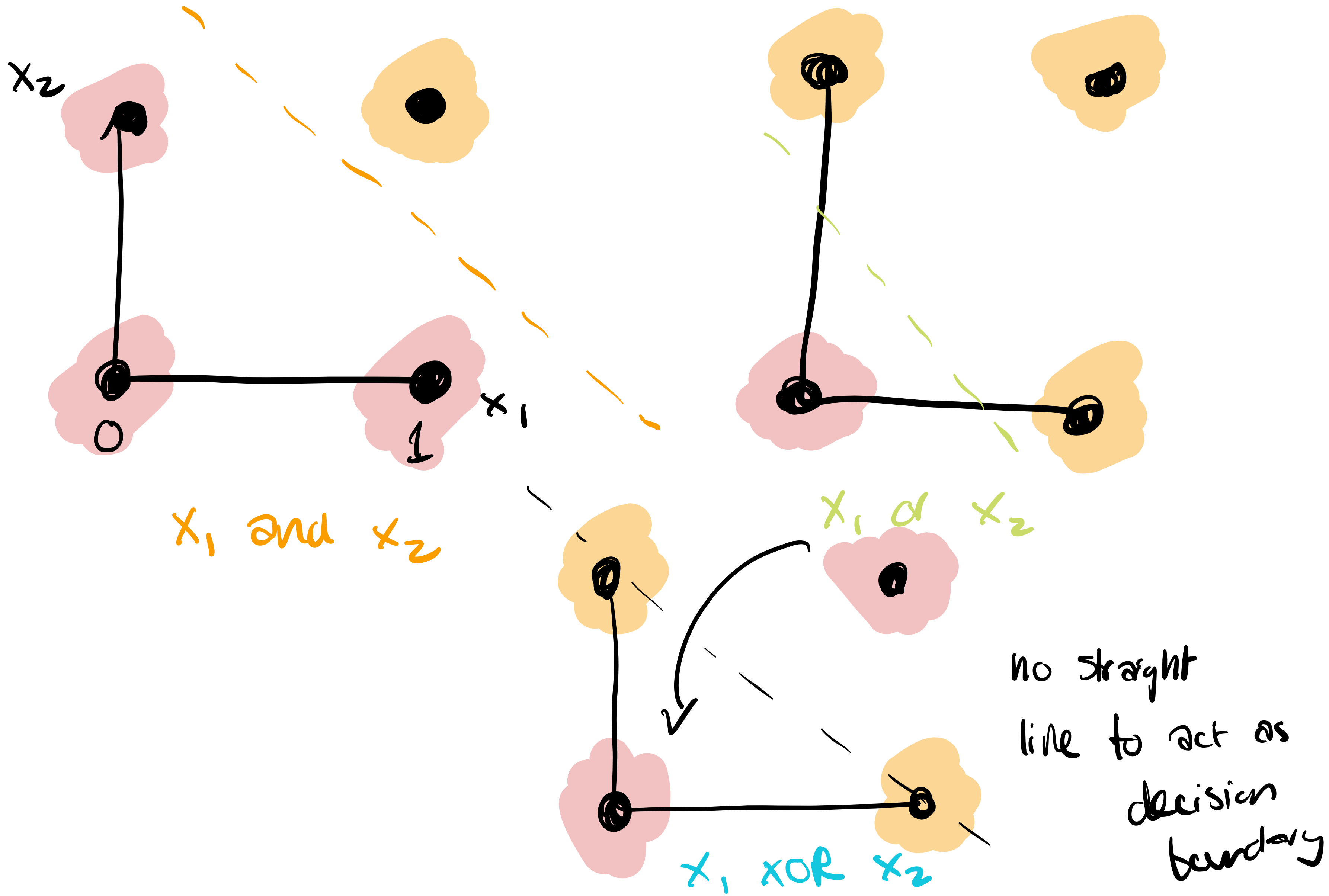
Perceptron equation is the equation of a line

$$w_1x_1 + w_2x_2 + b = 0$$

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

XOR can't be calculated by a single perceptron

XOR can be calculated by a layered network of units.

XOR

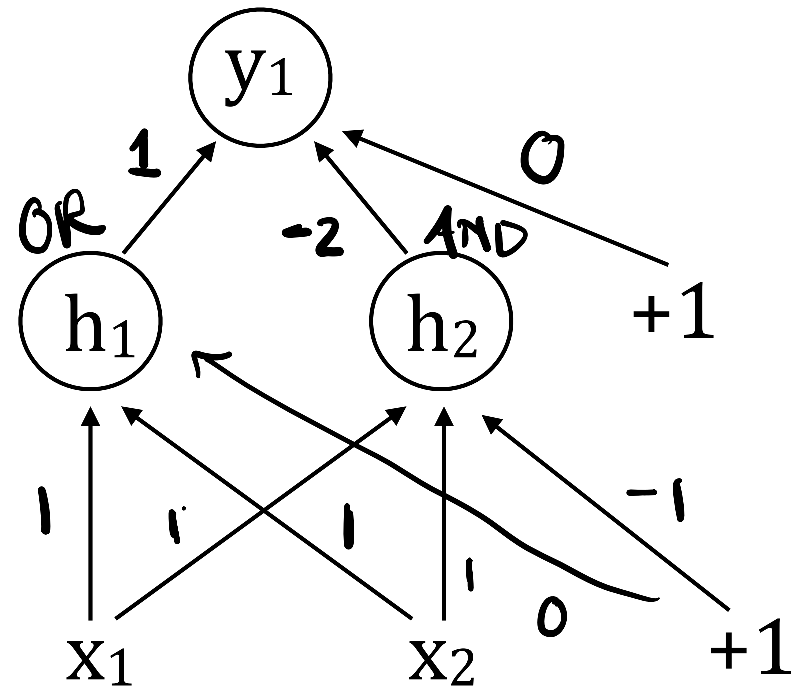
x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0

OR

$h_1 = x_1 + x_2$

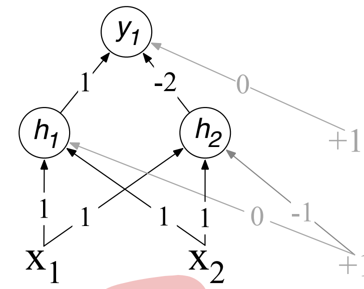
$h_2 = x_1 + x_2 - 1$

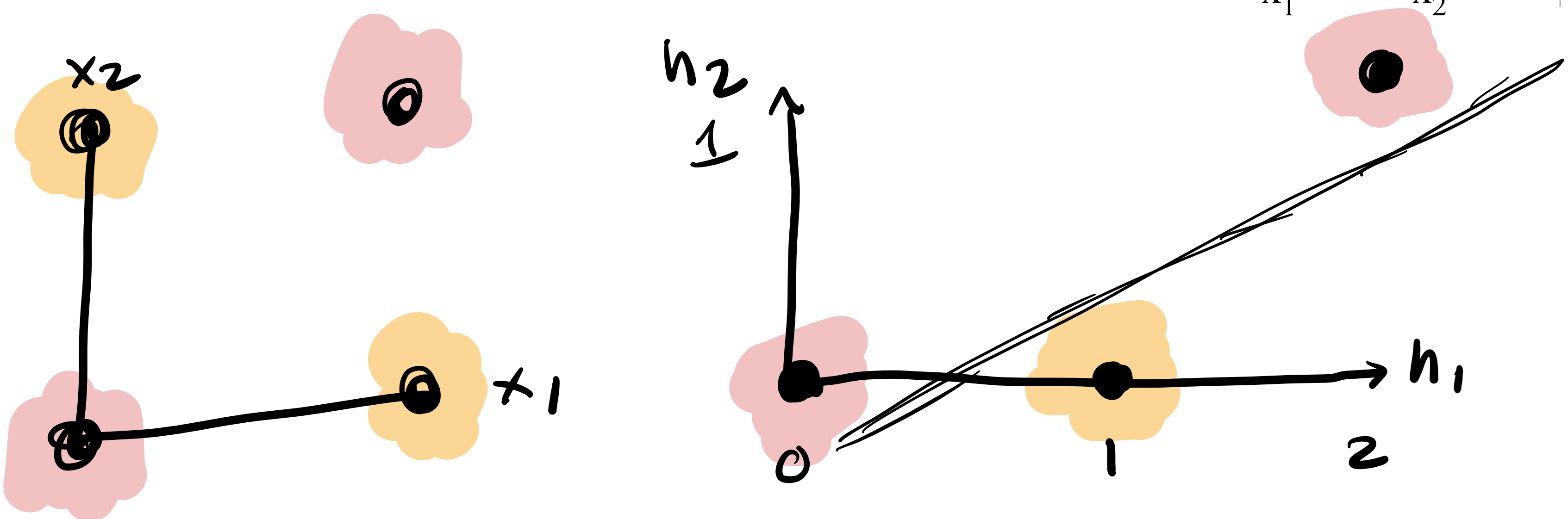
AND



$$h_2 = \text{ReLu}(w_{21}\, x_1 + w_{22}\, x_2 + 1)$$

The hidden representation h





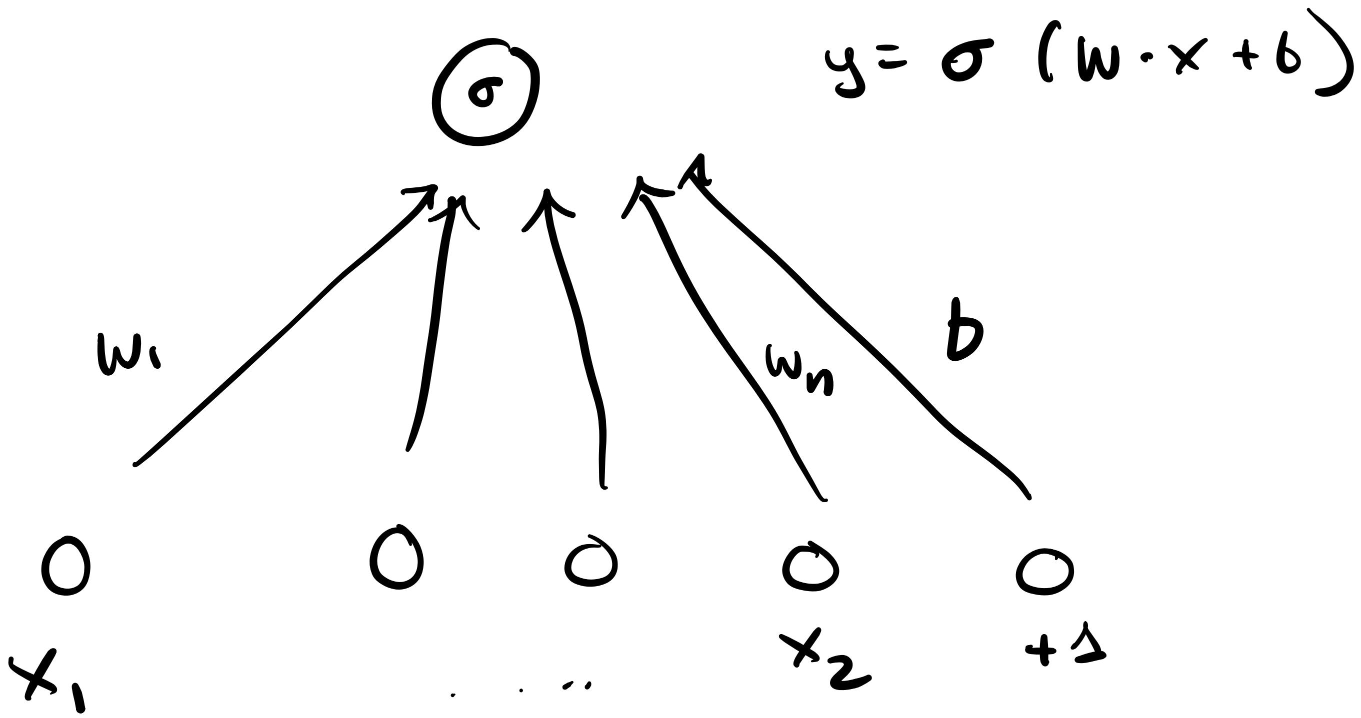
h_1 = OR

h_2 = AND

Feedforward Networks

Binary Logistic Regression as a 1-layer Network

(we don't count the input layer when counting layers!)

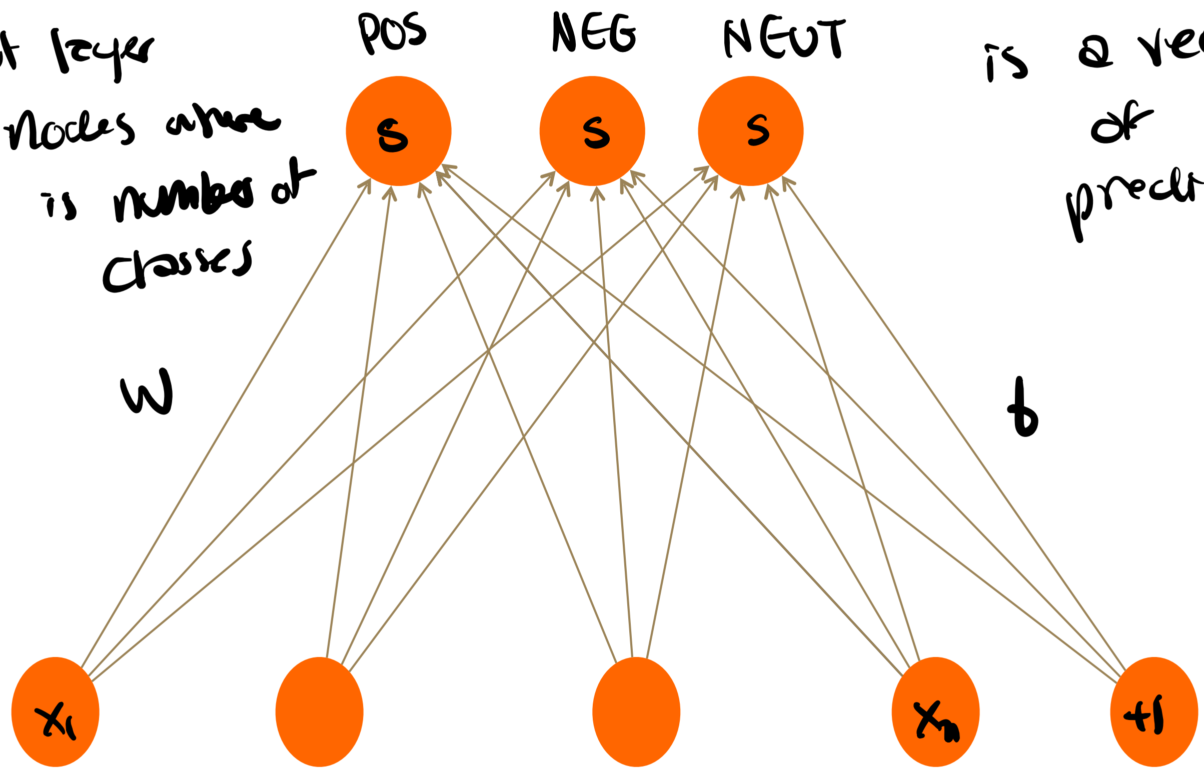


Multinomial Logistic Regression as a 1-layer Network

Fully connected single layer network

y = softmax(Wx+b) is a vector of predictions

Output layer has n nodes where n is number of classes



W is a matrix

n × X

X is a vector