INTRODUCTION TO MACHINE LEARNING COMPSCI 4ML3

Lecture 21

Hassan Ashtiani

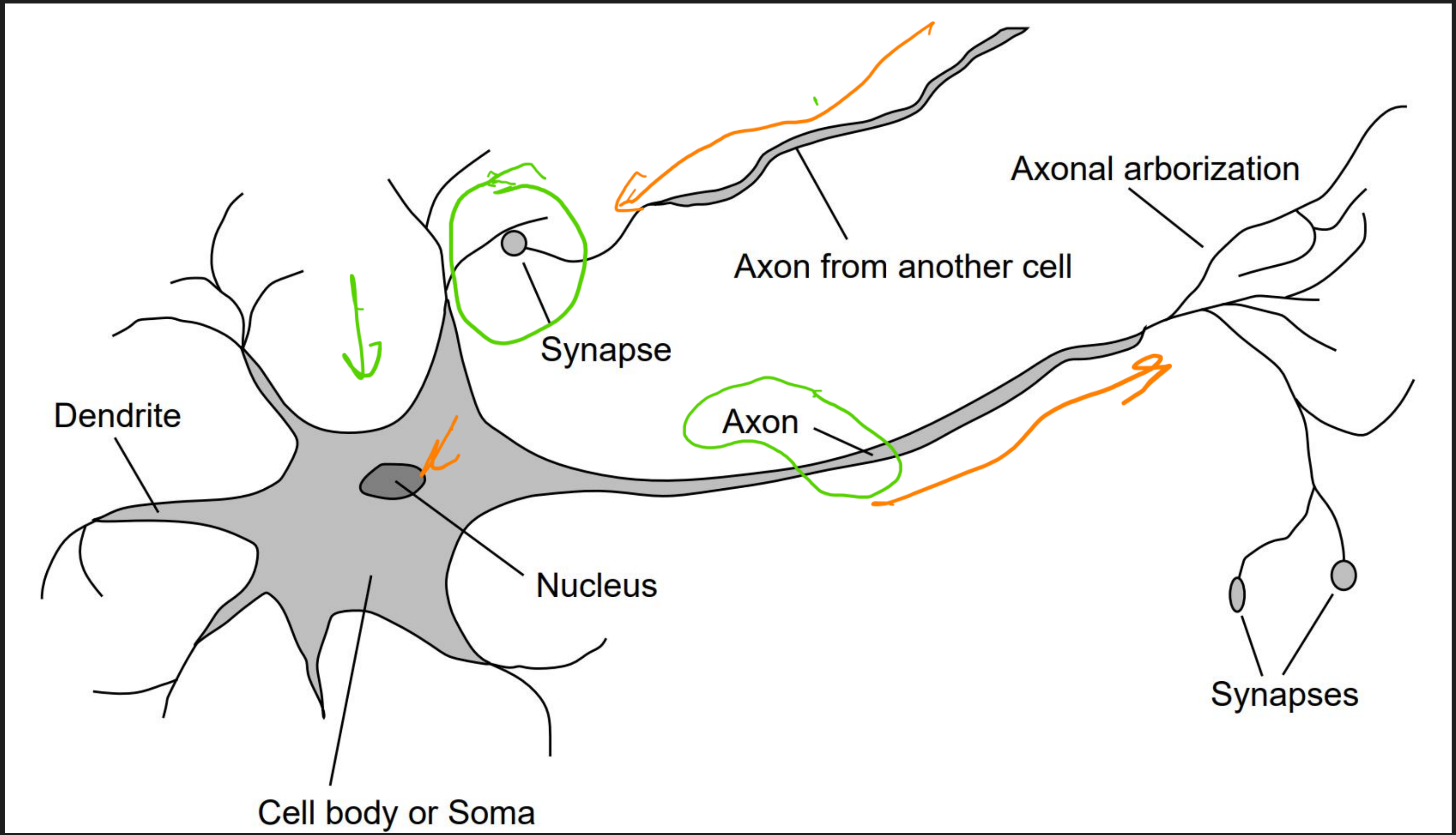
BRAIN VS COMPUTER PROGRAMS

- They both perform computations
- Computers are limited version of Turing Machines
  - Turing machines can do all the computations…
- Is there a points in mimicking/studying brain?
  - Computational Efficiency (memory, time, …)
  - Statistical efficiency (for real world data)

BRAIN VS COMPUTER PROGRAMS

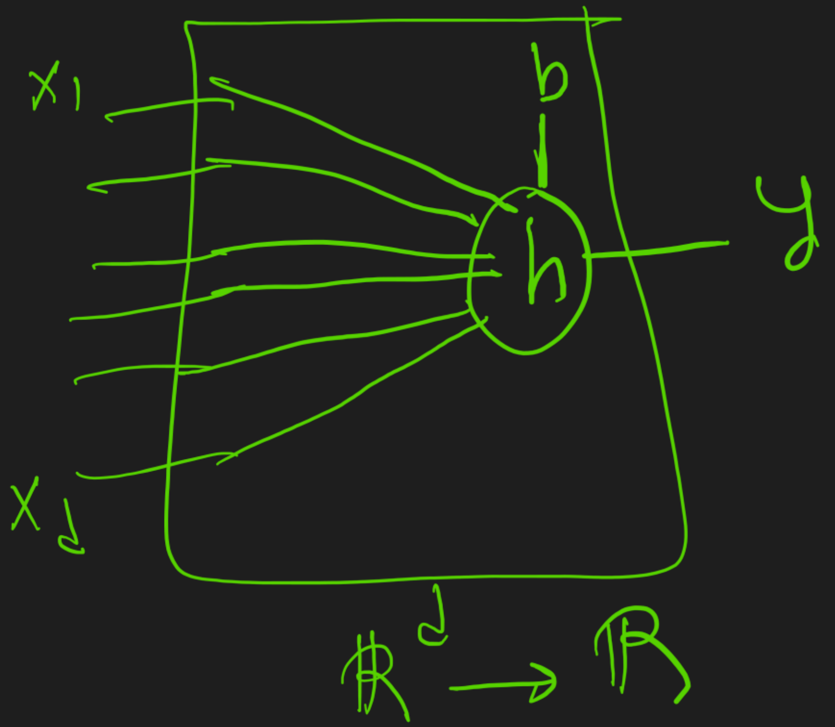
- Computer programs
  - Series of operations (mostly sequential)
  - Sensitive to change of the program
    - E.g., changing one line of code
- Connectionist Model (more like a circuit)
  - Network of neurons
  - Parallel computations
  - Robust (e.g., to removing one neuron)
- Still we can try to simulate brain with computers
  - Plus, specialized hardware like GPUs help!

NEURON



Axonal arborization
Axon from another cell
Synapse
Dendrite
Axon
Nucleus
Synapses
Cell body or Soma

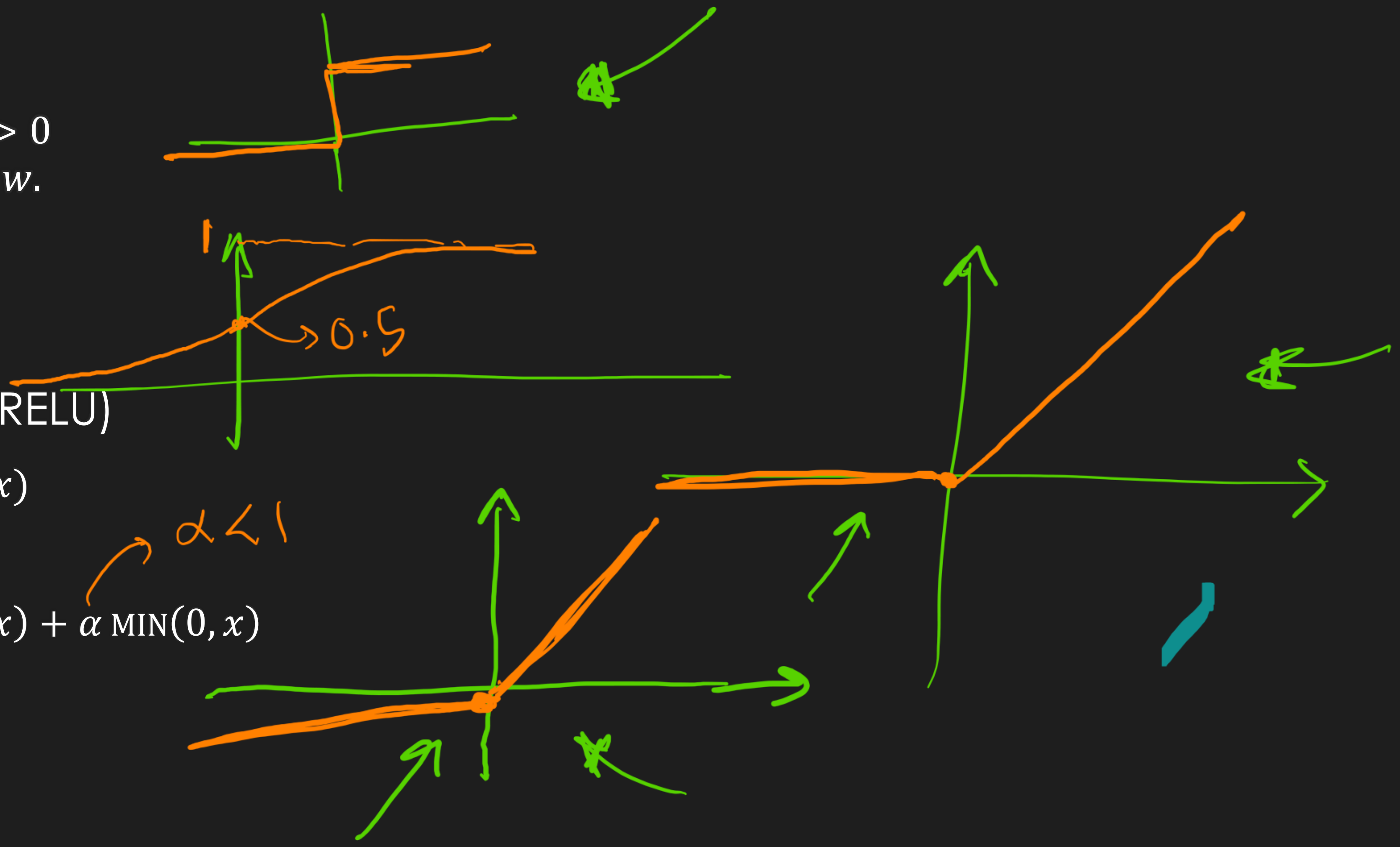
AN ARTIFICIAL NEURON



$$y = h(w^T x + b)$$

POPULAR ACTIVATION FUNCTIONS

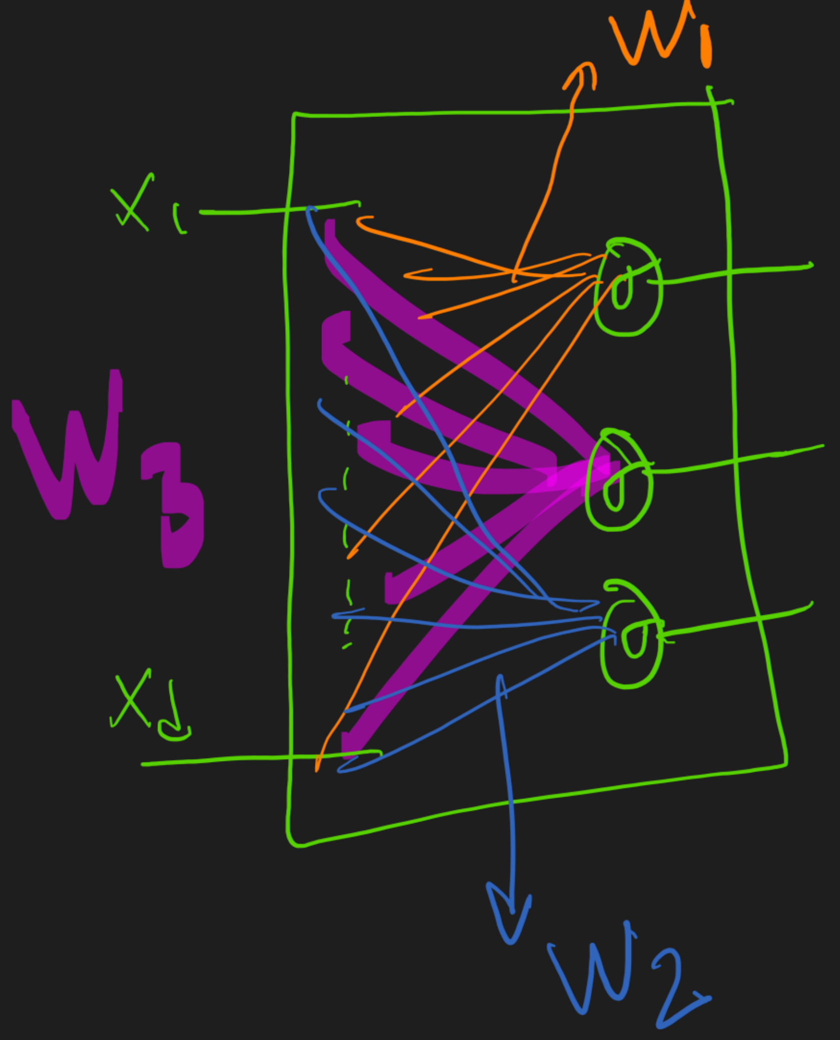
- THRESHOLD (SIGN)
  - $h(x) = \begin{cases} 1 & x > 0 \\ 0 & o.w. \end{cases}$
- SIGMOID
  - $h(x) = \frac{1}{1+e^{-x}}$
- RECTIFIED LINEAR (RELU)
  - $h(x) = \text{MAX}(0, x)$
- LEAKY RELU
  - $h(x) = \text{MAX}(0, x) + \alpha \, \text{MIN}(0, x)$ $\alpha < 1$
- TANH:
  - $h(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$

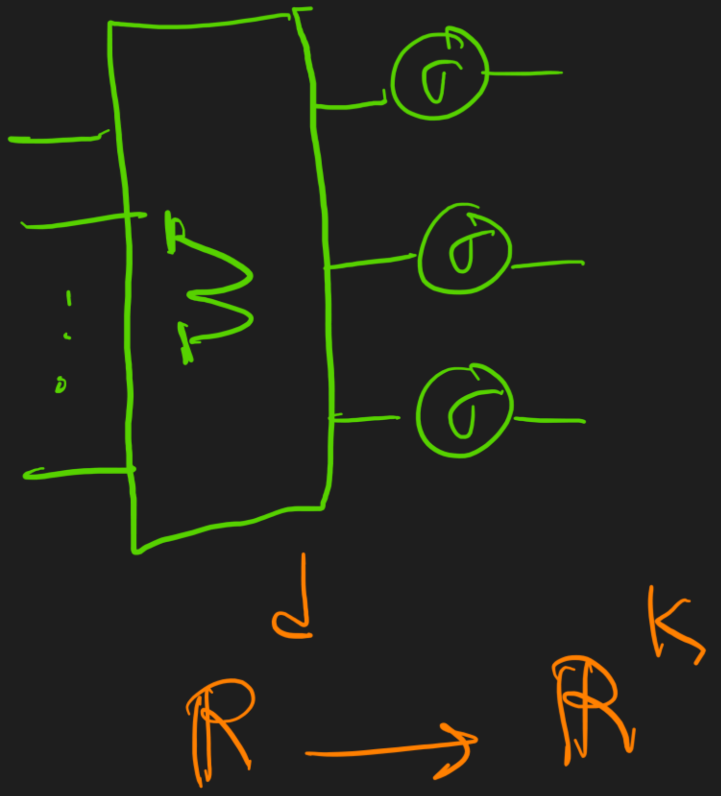


MULTIPLE OUTPUTS

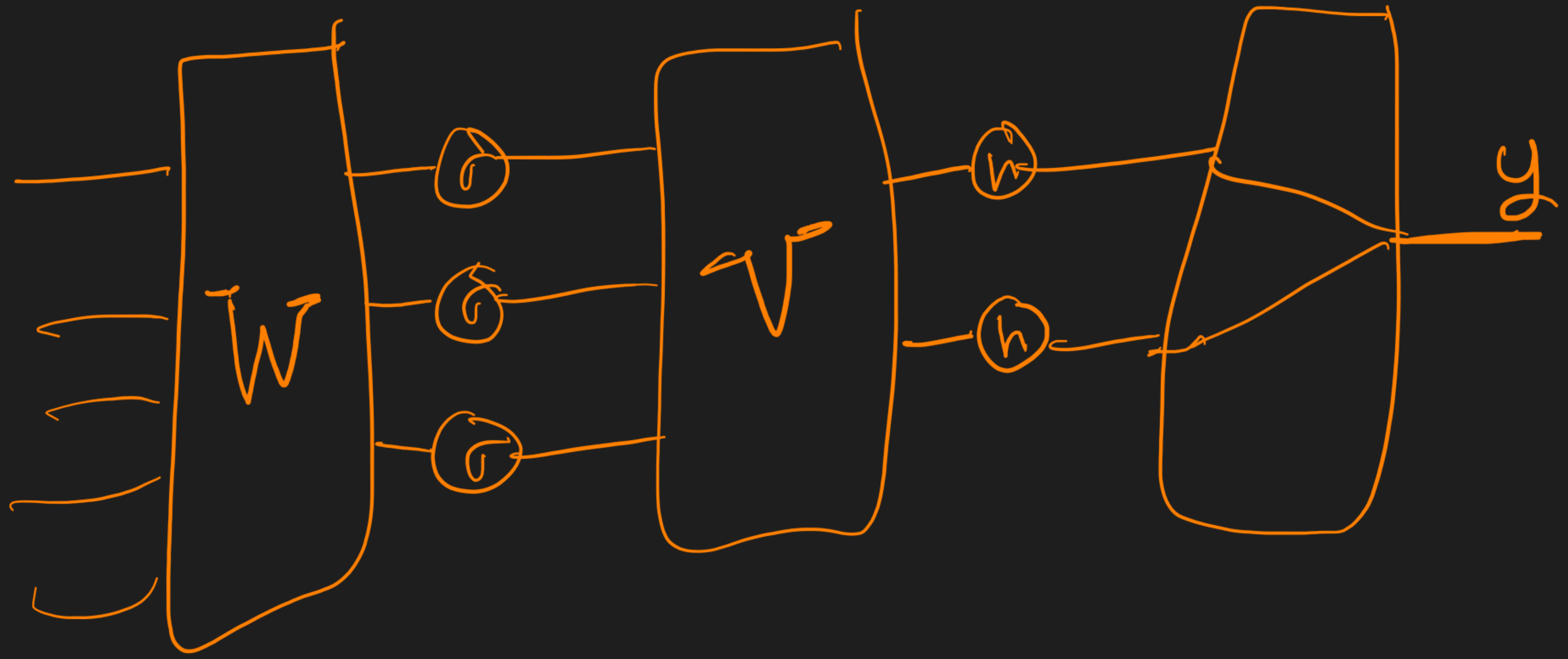
$$W \in \mathbb{R}^{d \times K}$$

$$W_{d\times K} = \begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix}_{d \times K}$$





FEED-FORWARD (VANILLA) NEURAL NETWORKS



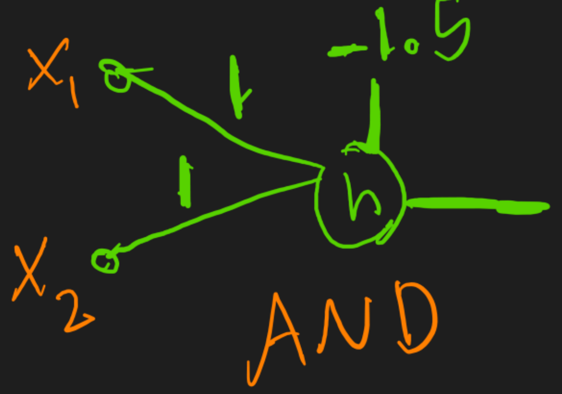
FEED-FORWARD NEURAL NETWORKS

- Feed-Forward models
  - Are Memoryless
  - Have No feedback loop
  - can be used for classification or regression (more on this later)
- Can we use linear activations functions?
  - The whole network will collapse to a linear function

EXAMPLE

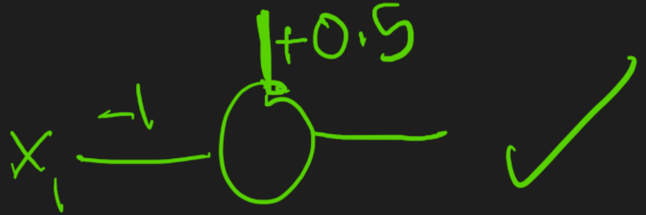
- Implement AND, OR, NOT, and XOR logical gates with neurons

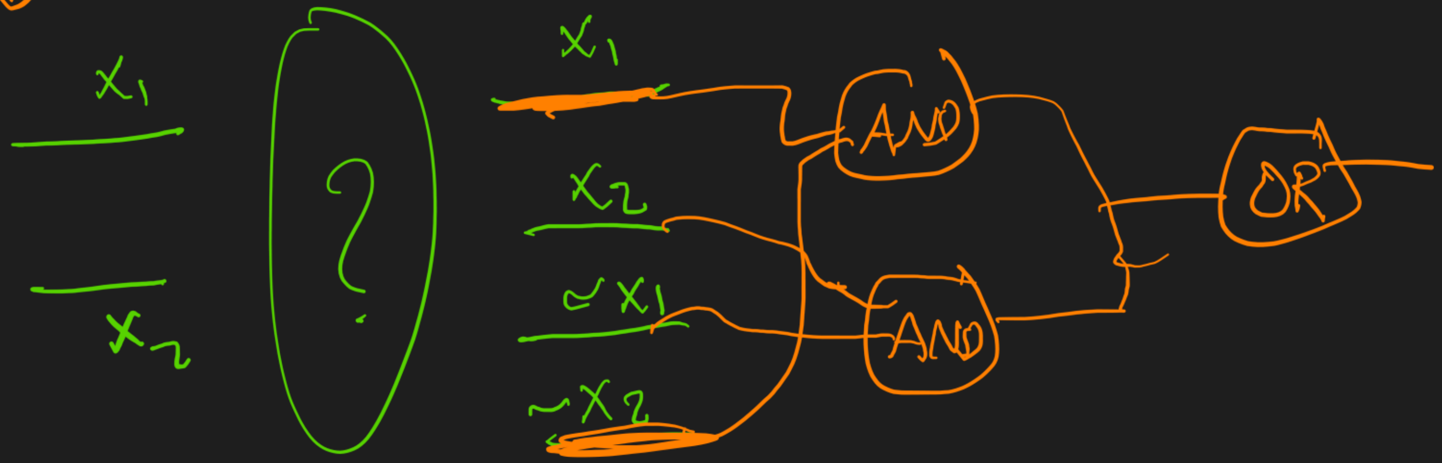
$x_1 \in \{0,1\}, \; x_2 \in \{0,1\}$



h: threshold activation

$h(x) = \mathbb{1}\{x > 0\}$





BOOLEAN FUNCTIONS

- Are neural networks powerful enough to represent any Boolean function (with finite inputs)?

UNIVERSAL APPROXIMATION THEOREM

- How flexible neural networks are when the input and output are continuous?
- Feed-forward networks with sigmoid activation functions can approximate any bounded continuous function up to desirable accuracy
  - **Only a single hidden layer is needed!**
  - George Cybenko, 1989
  - Also holds for other usual activation functions

UNIVERSAL APPROXIMATION THEOREM

- So are neural networks the best approach for learning?

* large neural nets require
  * A lot of training data
  * A lot of compute

sometimes other models are more efficient.