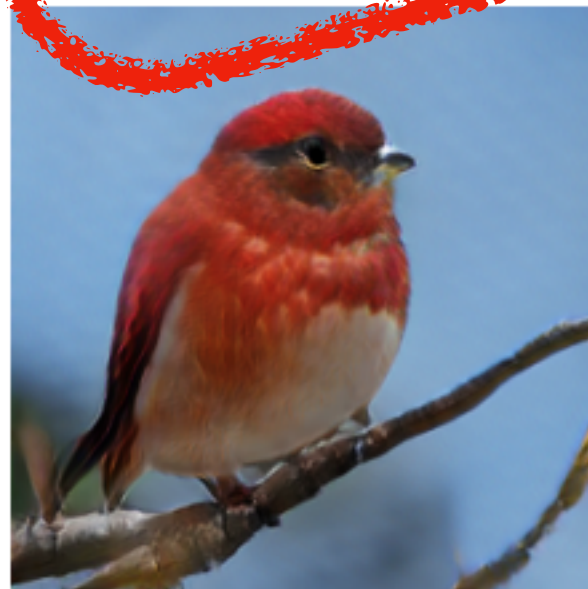
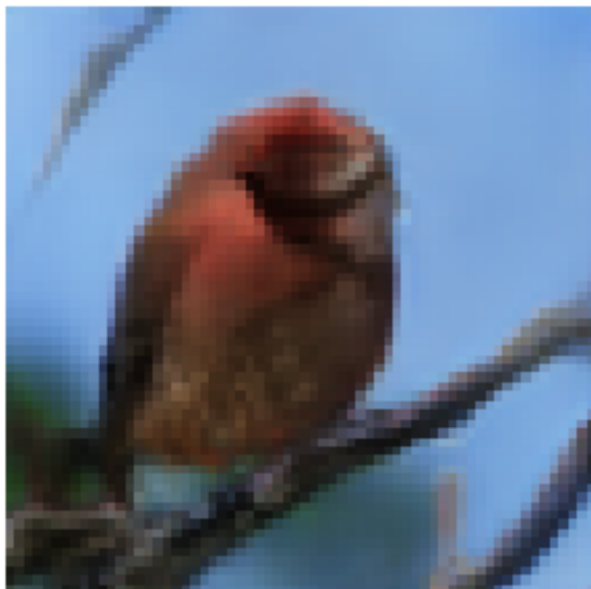


Neural Nets Everywhere



**FAKE
NEWS**

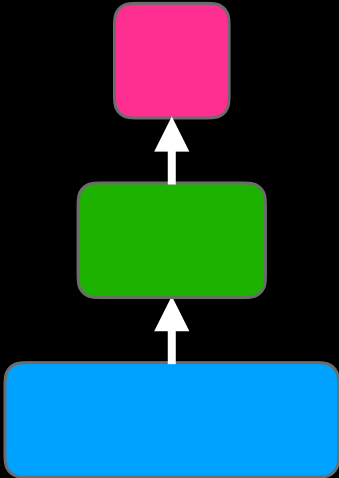
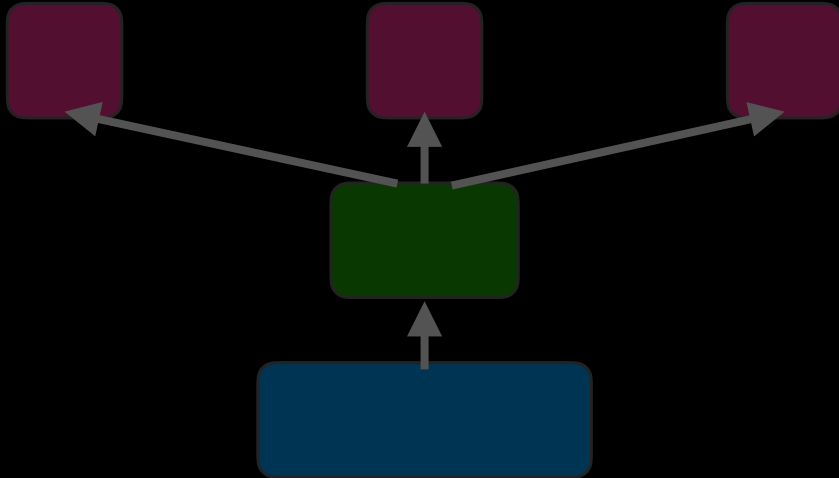
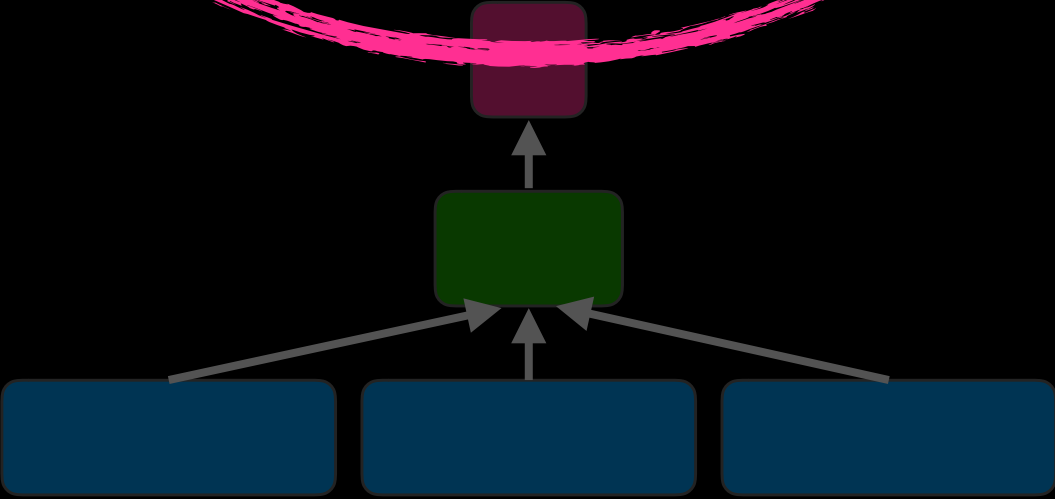
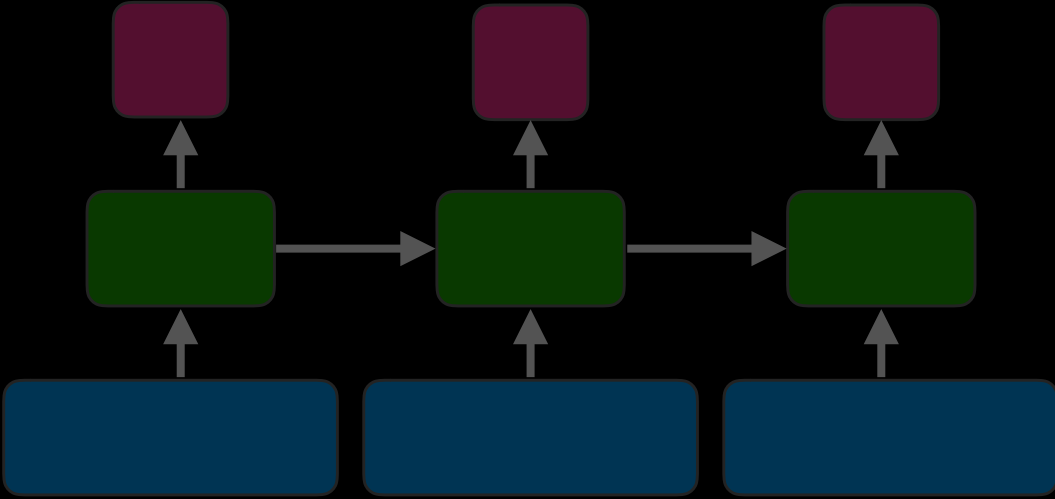
this bird is red with white and has a very short beak



Goals for Today

- Learn the basic difference of **neural architectures**
- Understand the **perceptron** as a basic element
- Understand training through **backpropagation**
- Learn about **dropout regularization**

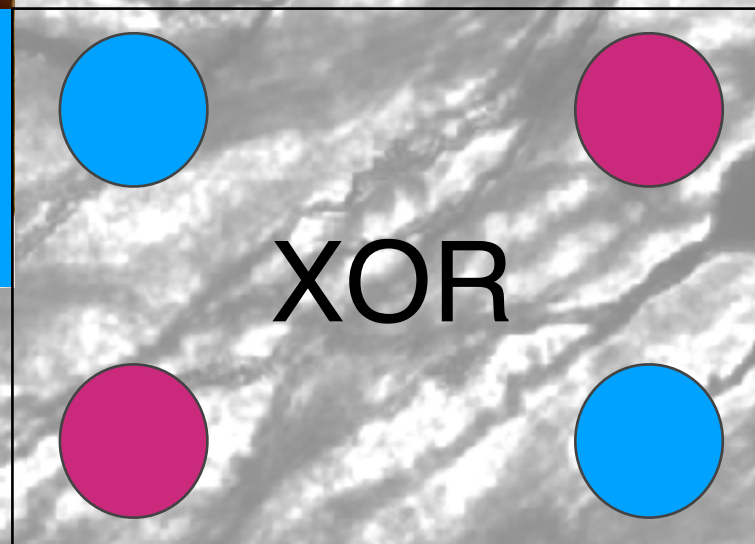
Types of Neural Models

	Fixed length	Variable length
Fixed length	 <p>Logistic Regression, Perceptron, Feed-Forward Network, Deep Belief Network...</p>	 <p>Multitask Learning, Decoder</p>
Variable length	 <p>Convolutional Neural Networks (CNN)</p>	 <p>Recurrent Neural Networks (RNN), Hidden Markov Models (HMM), Conditional Random Fields</p>



Frank Rosenblatt
(1928–1971)

The Perceptron can learn
anything!!!



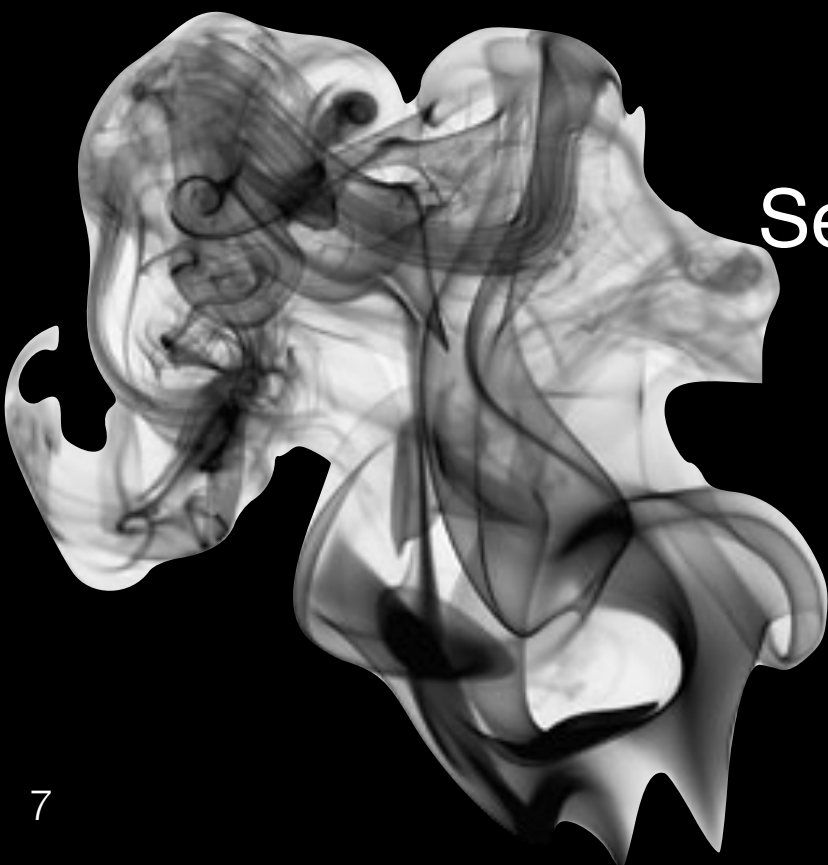
The Perceptron fails at learning
even basic concepts



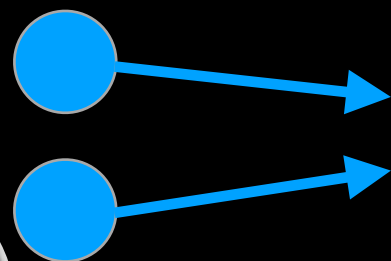
Marvin Minsky
(1927–2016)

The Perceptron

A Threshold Unit



Sensor array



Total smoke

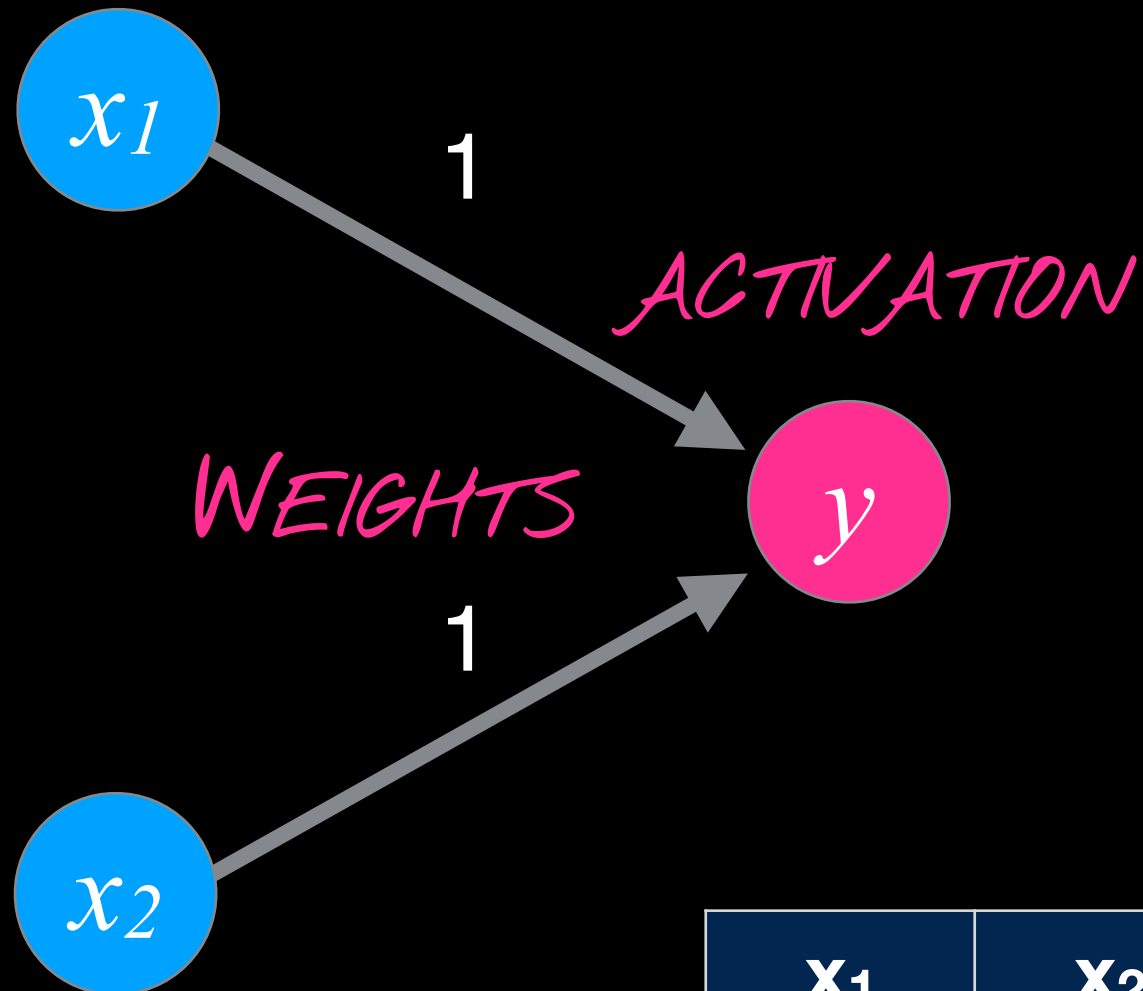
if $>$ threshold



Bocconi

OR-Perceptron

INPUT



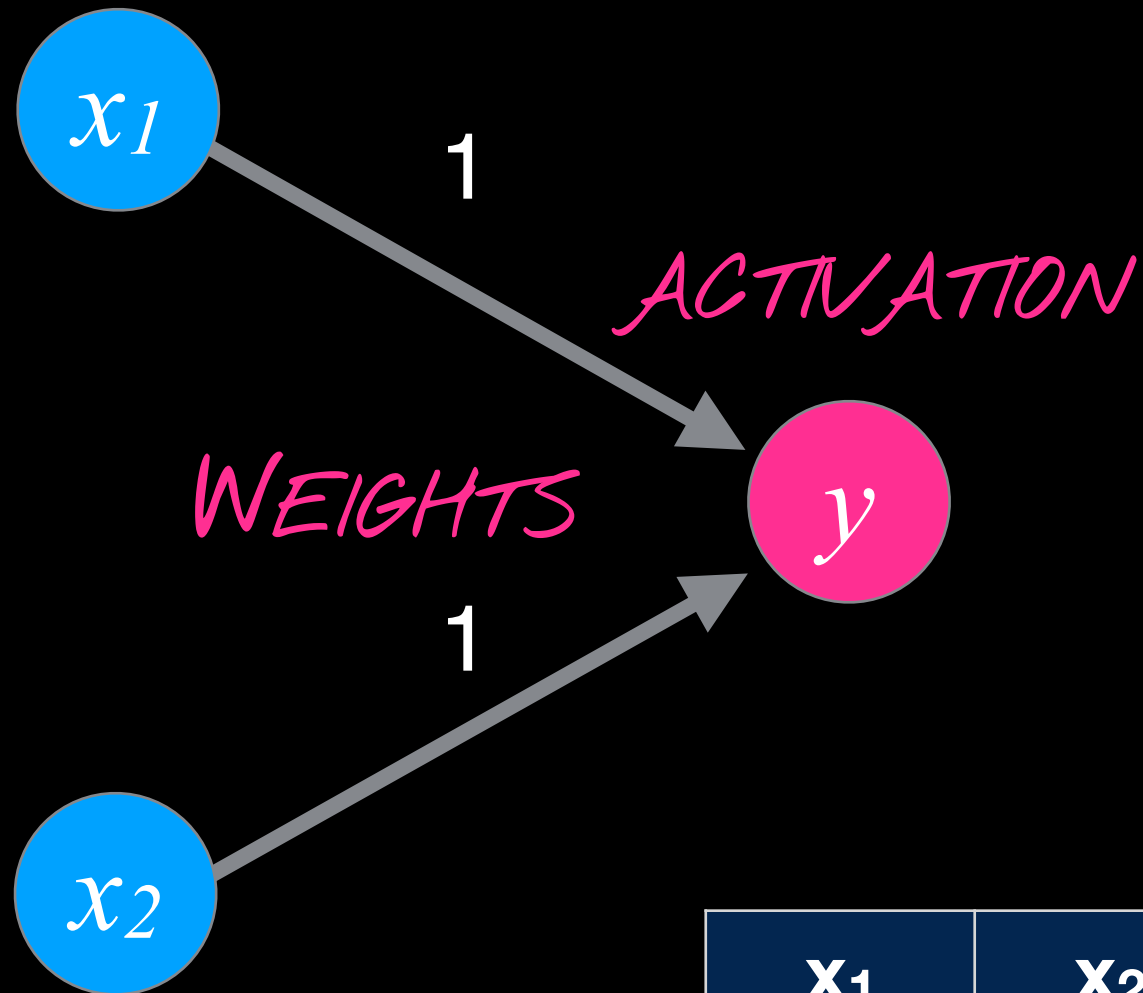
$$f(X) = w_1 x_1 + w_2 x_2$$

$$\hat{y} = \begin{cases} +1 & \text{if } f(X) \geq 1 \\ -1 & \text{otherwise} \end{cases}$$

x_1	x_2	y
0	0	-1
1	0	1
0	1	1
1	1	1

AND-Perceptron

INPUT

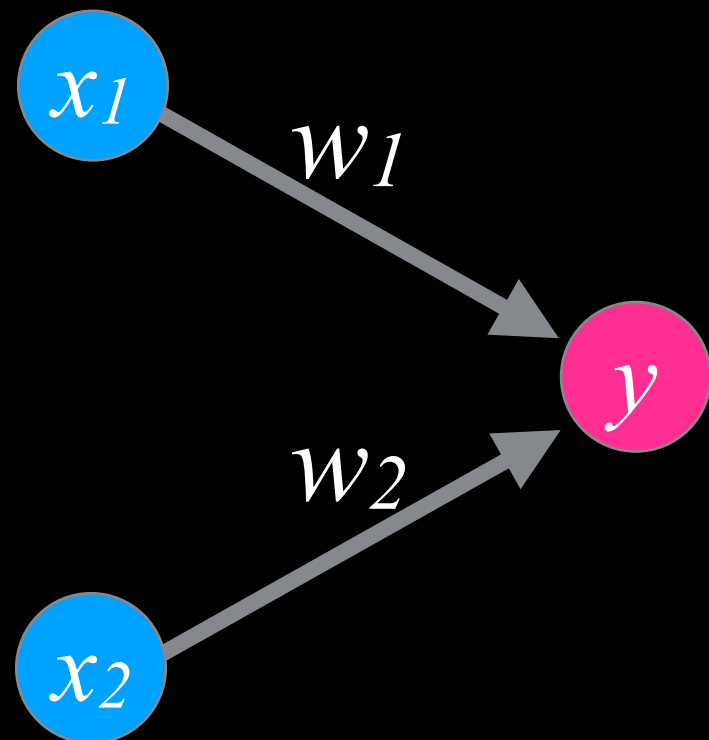


$$f(X) = w_1 x_1 + w_2 x_2$$

$$\hat{y} = \begin{cases} +1 & \text{if } f(X) \geq 2 \\ -1 & \text{otherwise} \end{cases}$$

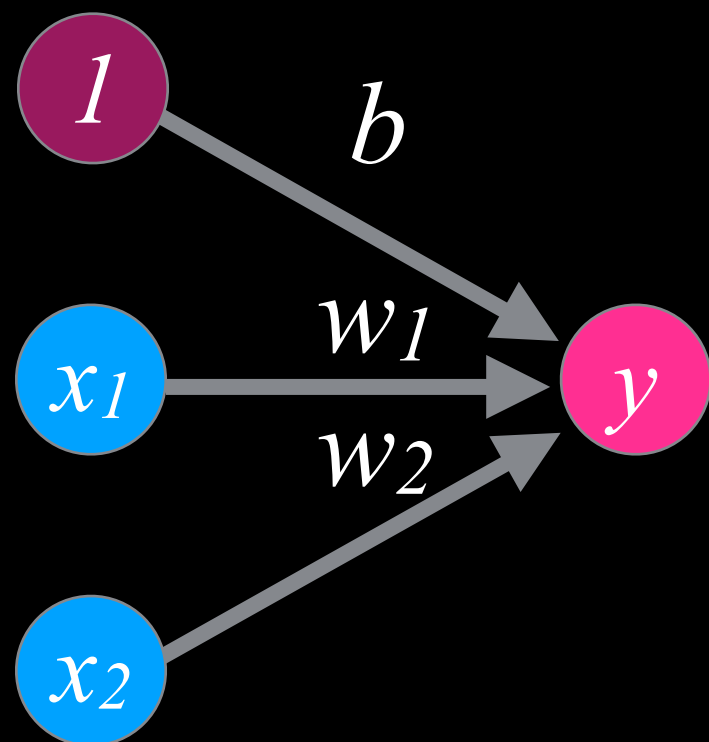
x_1	x_2	y
0	0	-1
1	0	-1
0	1	-1
1	1	1

Learn the Threshold



$$f(X) = w_1 x_1 + w_2 x_2$$

$$\hat{y} = \begin{cases} +1 & \text{if } f(X) \geq t \\ -1 & \text{otherwise} \end{cases}$$



$$f(X) = w_1 x_1 + w_2 x_2 + \textcolor{purple}{b}$$

$$\hat{y} = \begin{cases} +1 & \text{if } f(X) \geq \textcolor{purple}{0} \\ -1 & \text{otherwise} \end{cases}$$

Learning to Distinguish

$$f(X) = 1x_1 + 1x_2 - 1$$

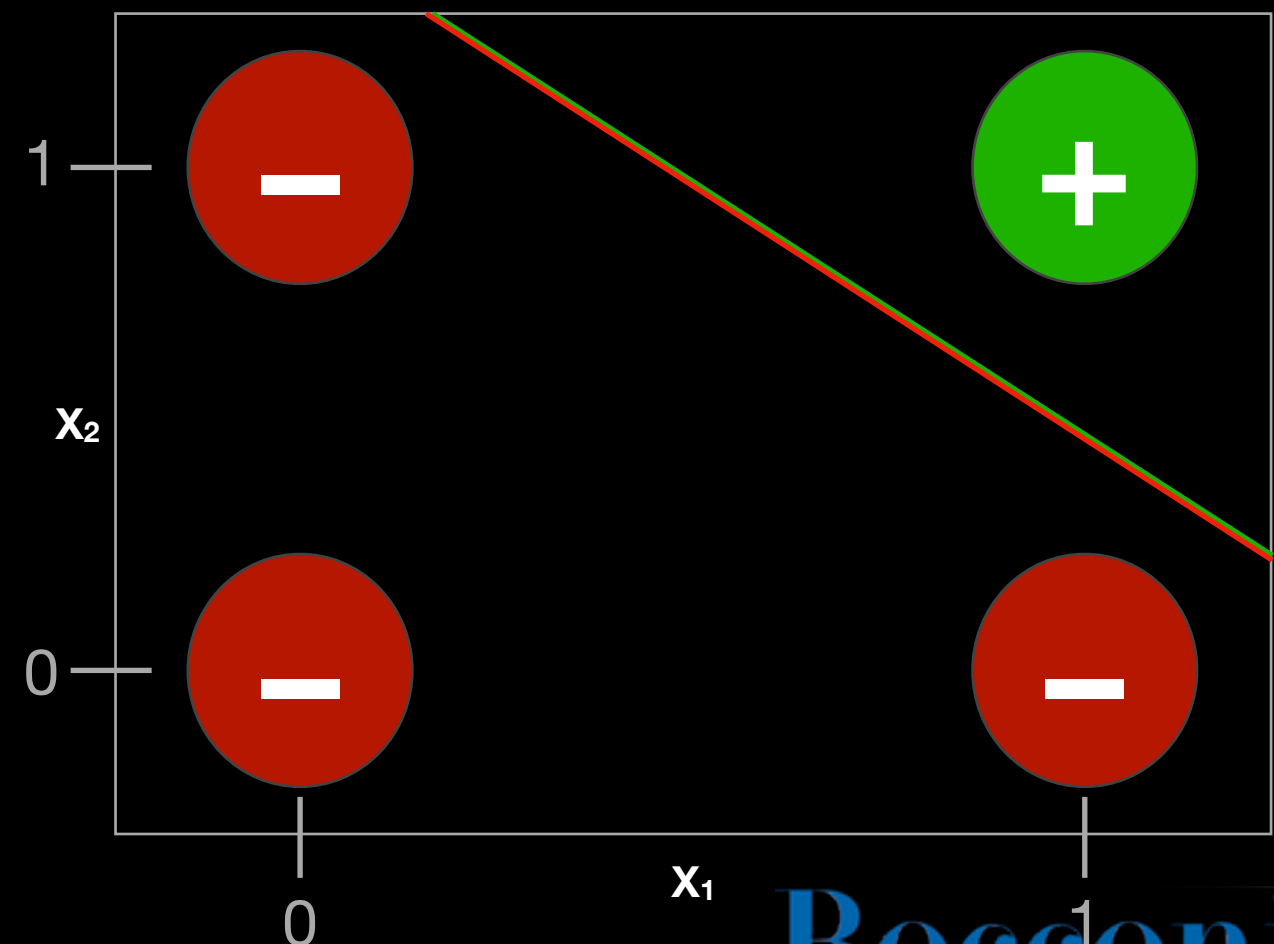
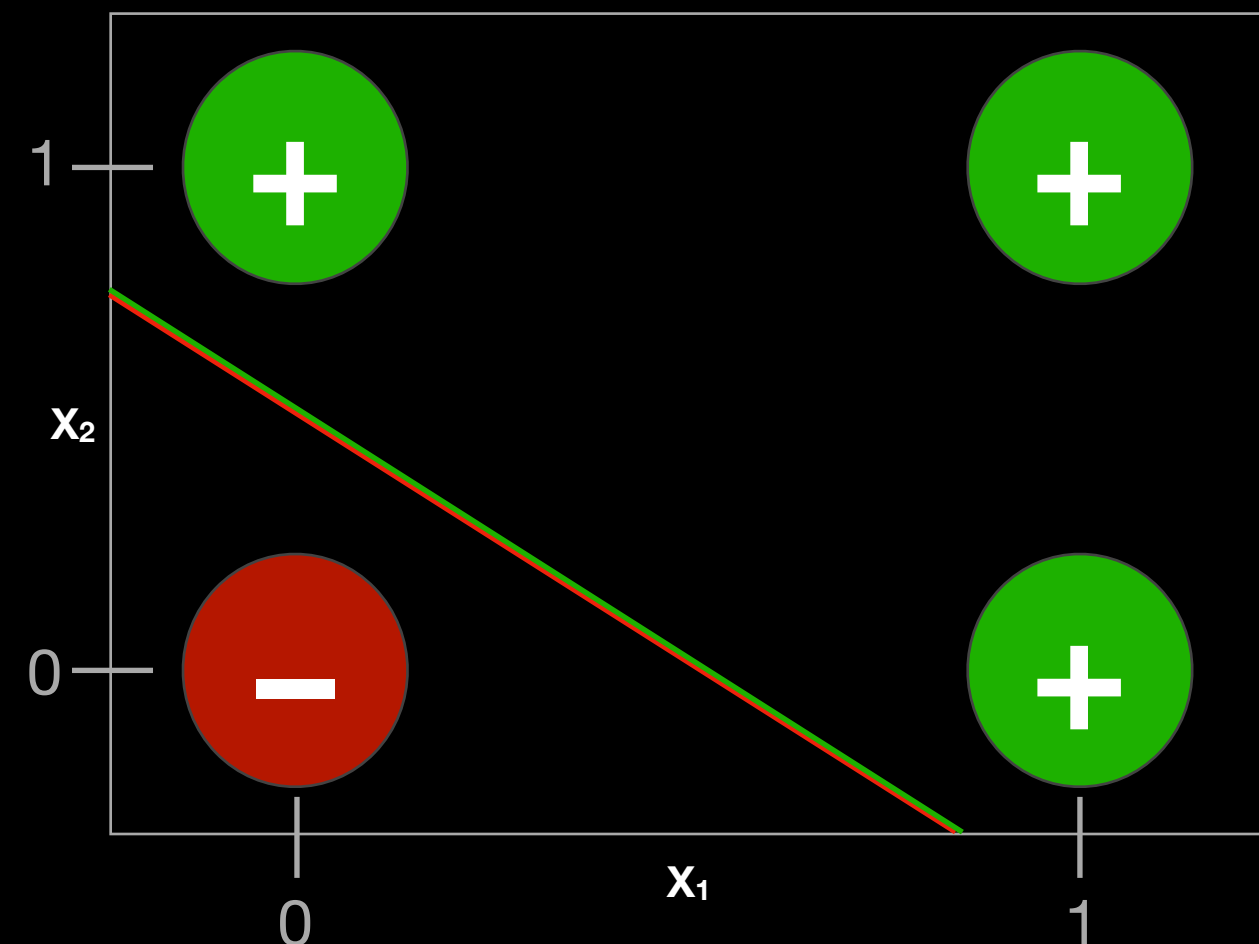
OR

x_1	x_2	y
0	0	-1
1	0	1
0	1	1
1	1	1

$$f(X) = 1x_1 + 1x_2 - 2$$

AND

x_1	x_2	y
0	0	-1
1	0	-1
0	1	-1
1	1	1



Decision Boundary

book (+1)



magazine (-1)



thickness

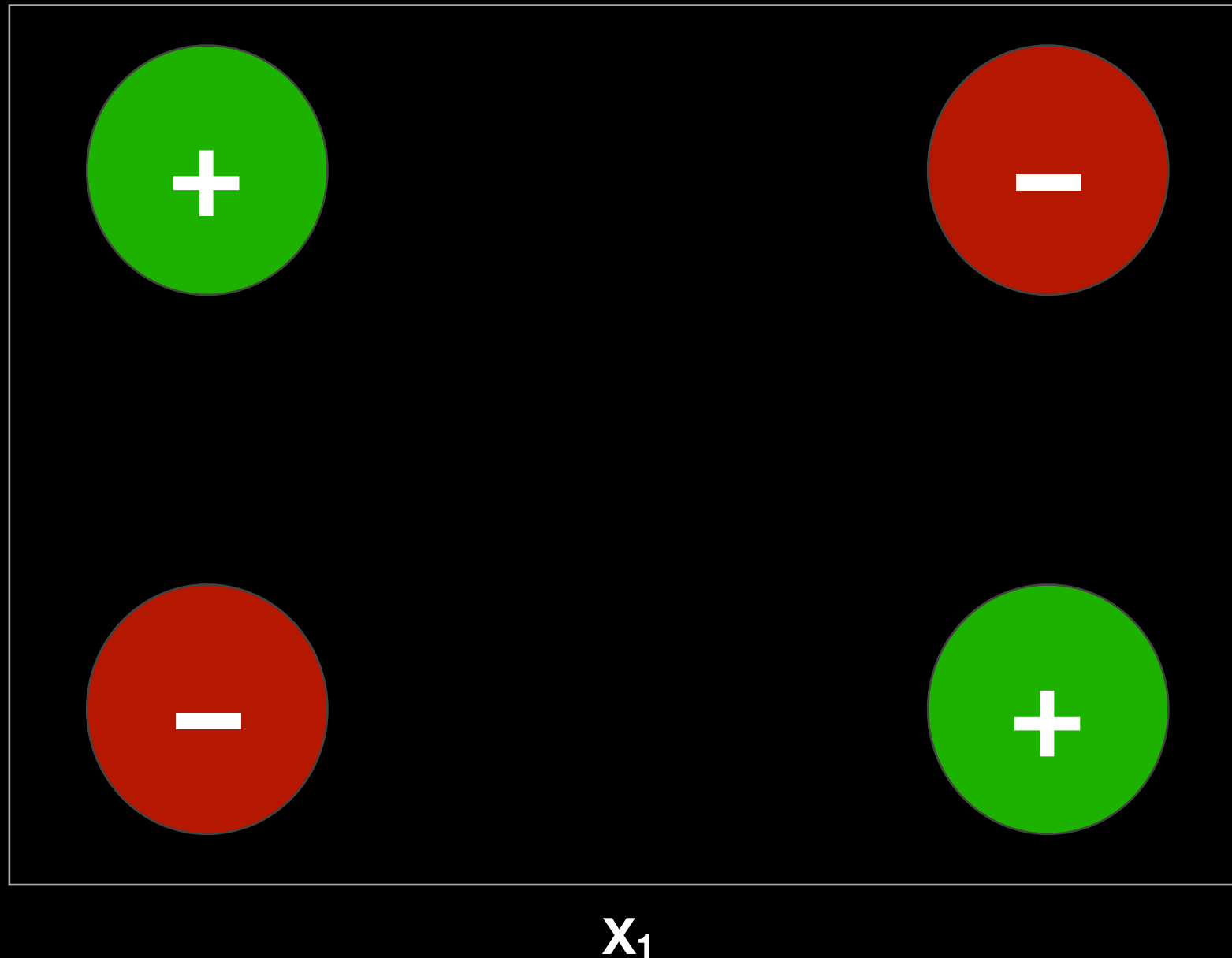
size

Wx_i

The XOR Limit

x_1	x_2	y
0	0	-1
1	0	1
0	1	1
1	1	-1

LINEARIZE THIS!

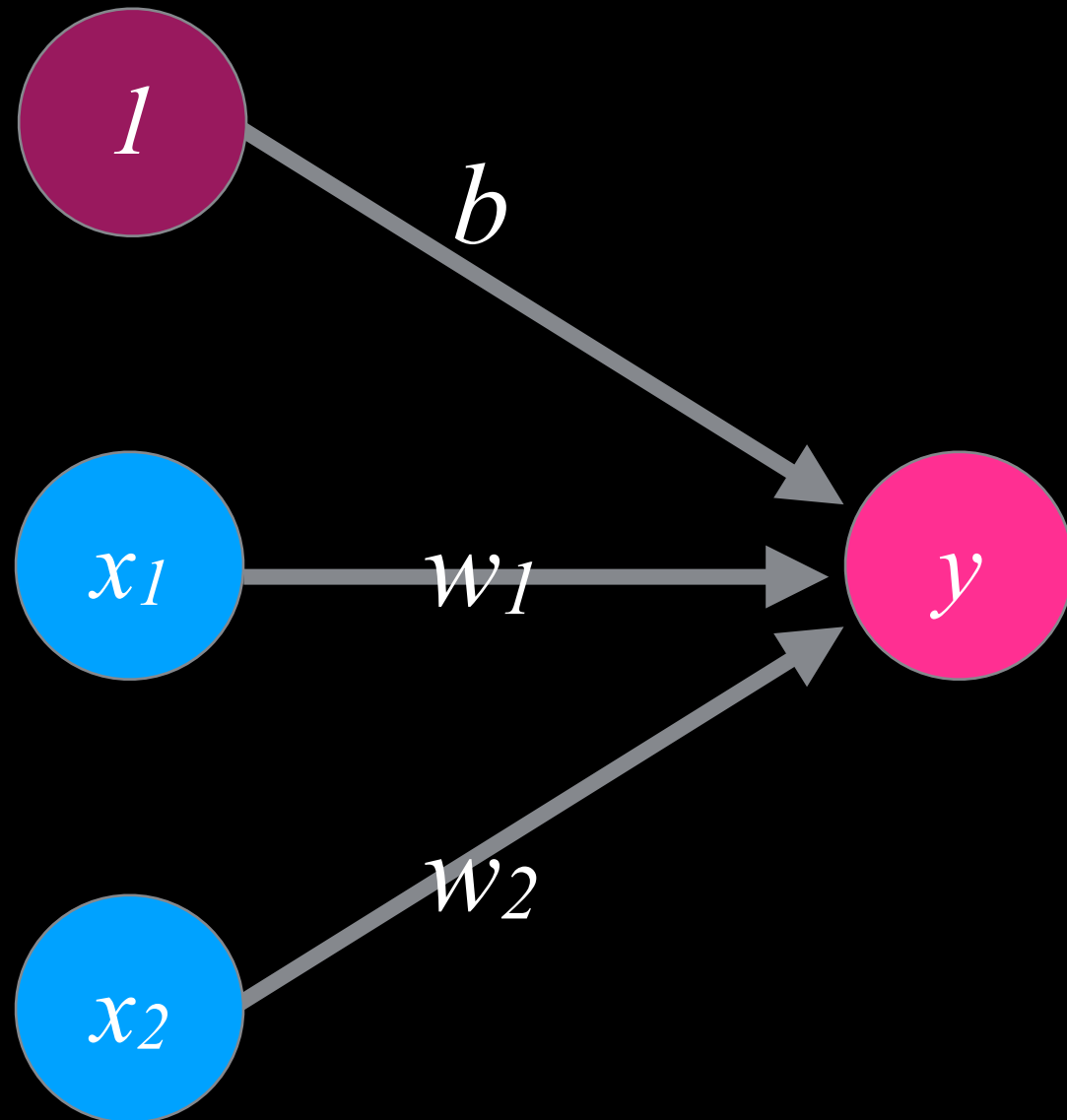


Marvin Minsky
(1927–2016)

Step 1: Non-Linearity

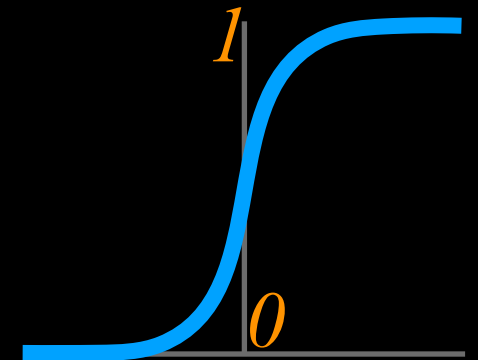
Nonlinear Activation Functions

$$f(X) = a(w_1 x_1 + w_2 x_2 + b)$$



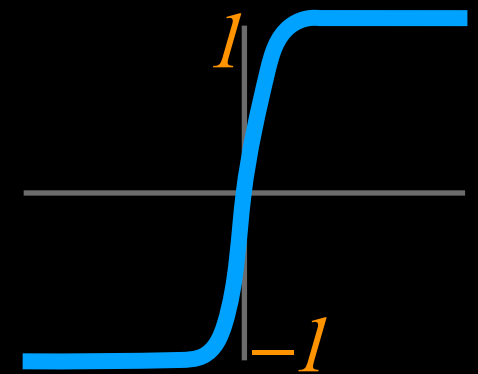
Sigmoid

$$s(x) = \frac{1}{1+e^{-x}}$$



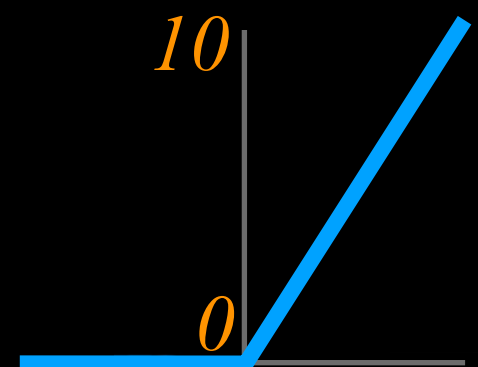
tanh

$$\tanh(x)$$



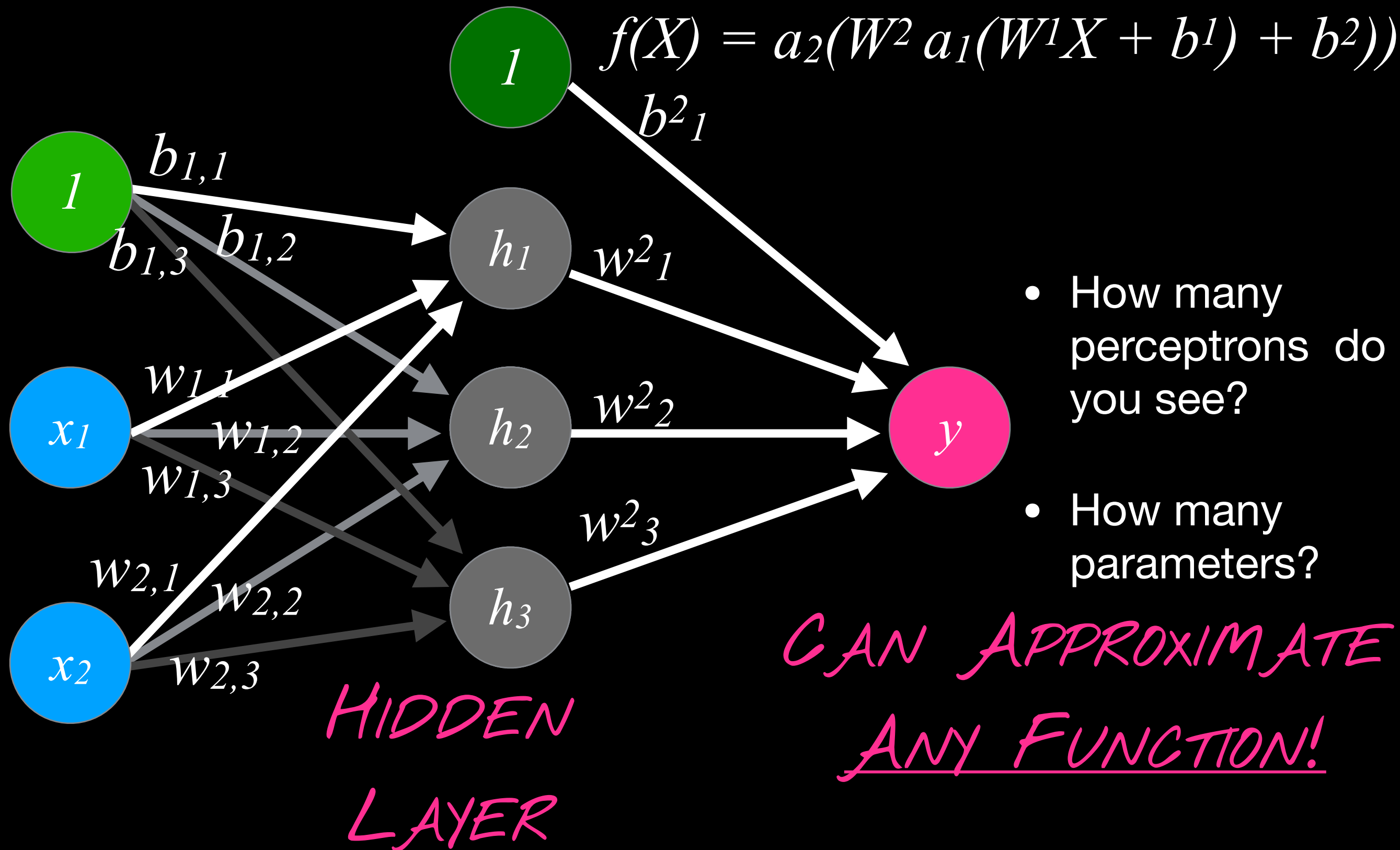
ReLU

$$\max(0, x)$$



Step 2: Going Deep – The Multilayer Perceptron

Multilayer Perceptron



- How many perceptrons do you see?
- How many parameters?

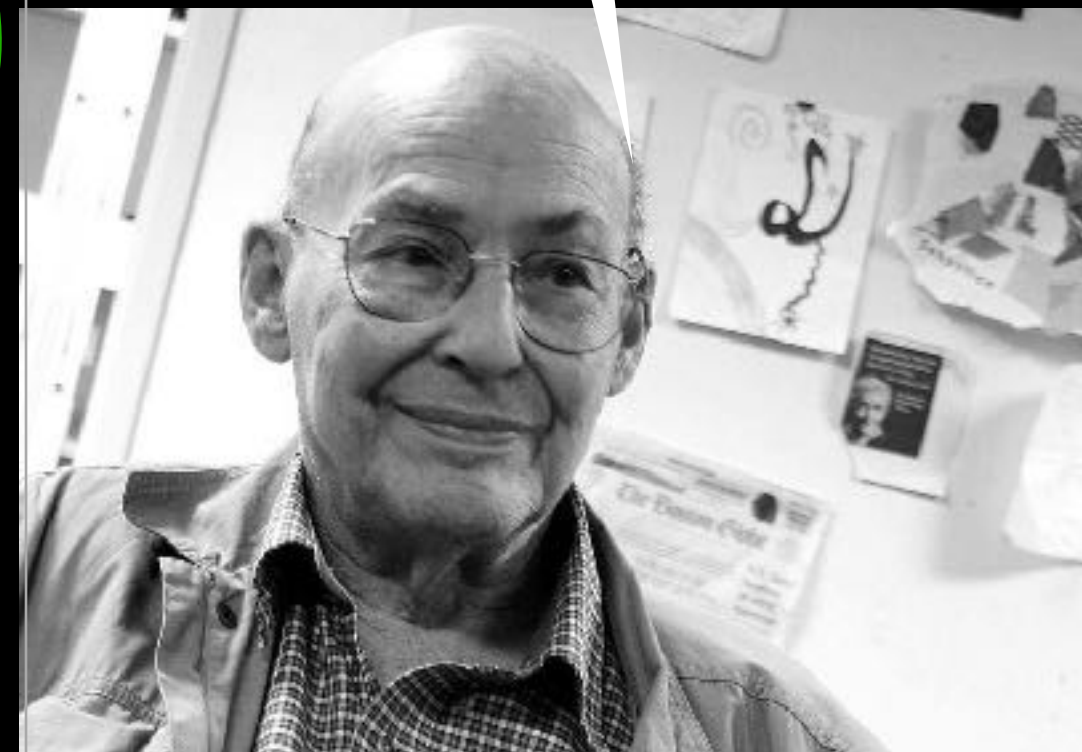
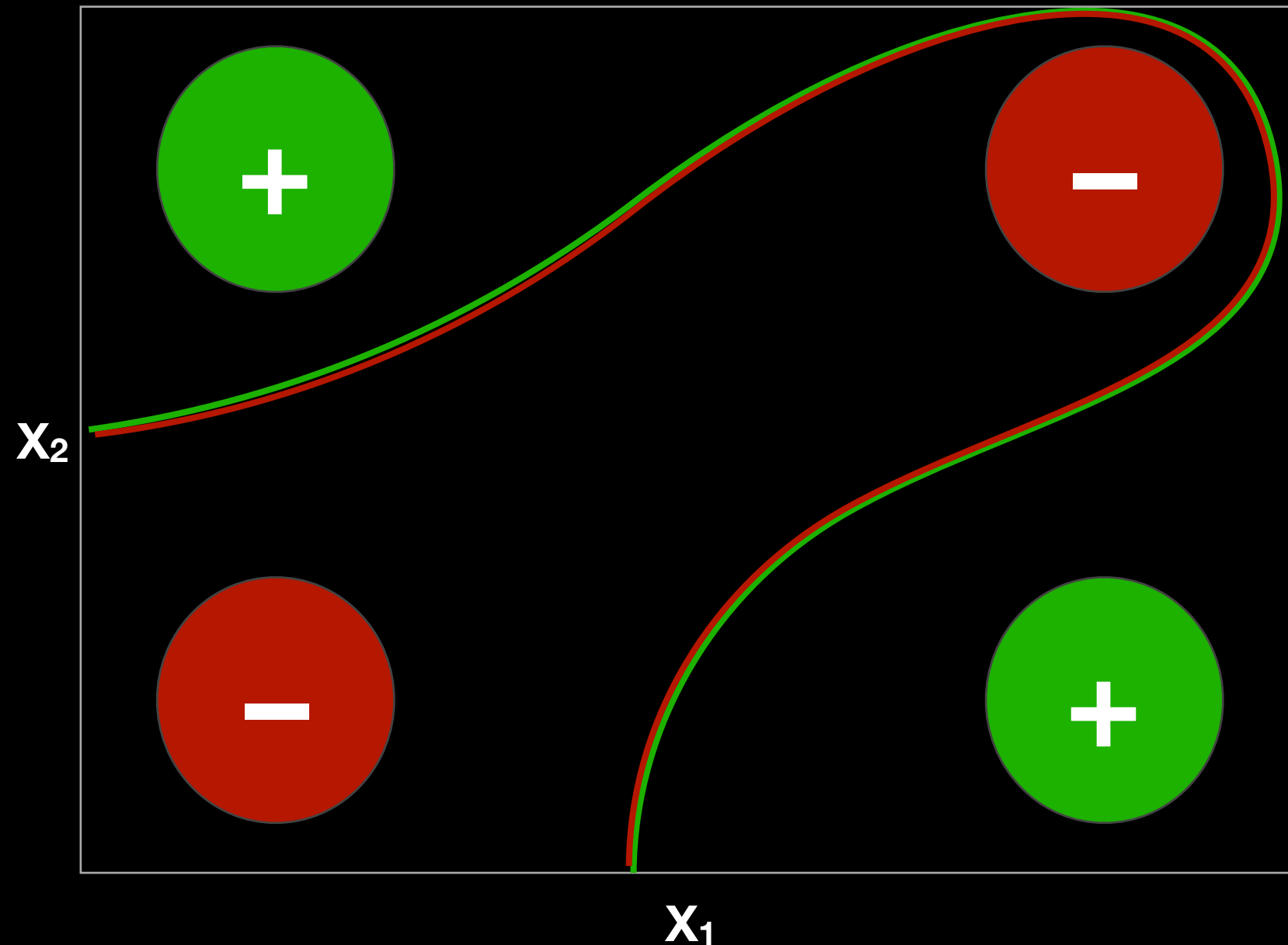
*CAN APPROXIMATE
ANY FUNCTION!*

The XOR Limit

x_1	x_2	
0	0	0
1	0	0
0	1	1
1	1	-1

THERE YOU GO...

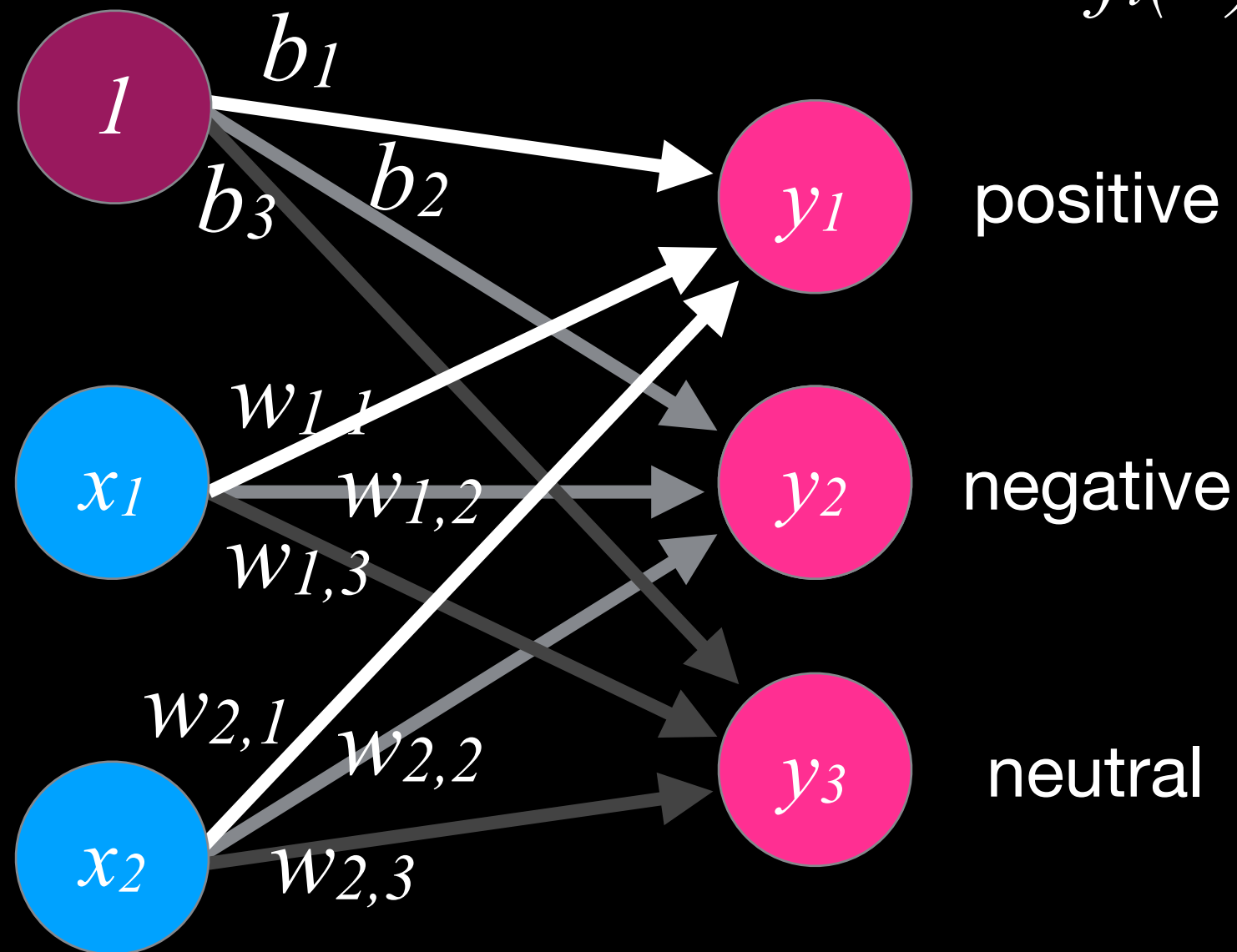
LINEARIZE THIS!



Marvin Minsky
(1927–2016)

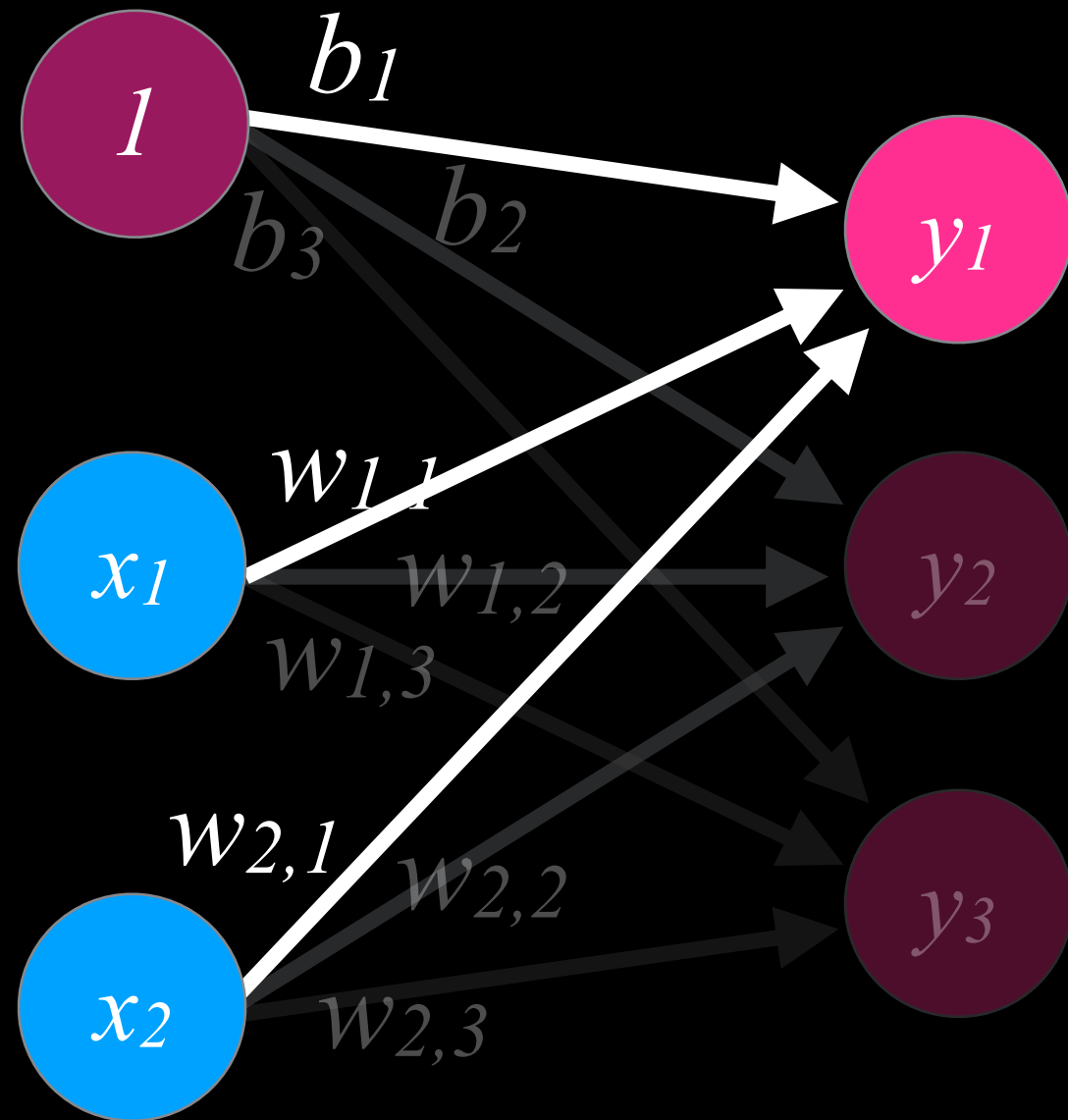
Multi-Class Output

$$f_i(X) = a(w_{1,i}x_1 + w_{2,i}x_2 + b_i)$$



$$\hat{y} = \underset{i}{\operatorname{argmax}} f_i(X)$$

Enter the Matrix



$$y_1 = a(w_{1,1}x_1 + w_{2,1}x_2 + b_1)$$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} w_{1,1} & w_{2,1} \\ w_{1,2} & w_{2,2} \\ w_{1,3} & w_{2,3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

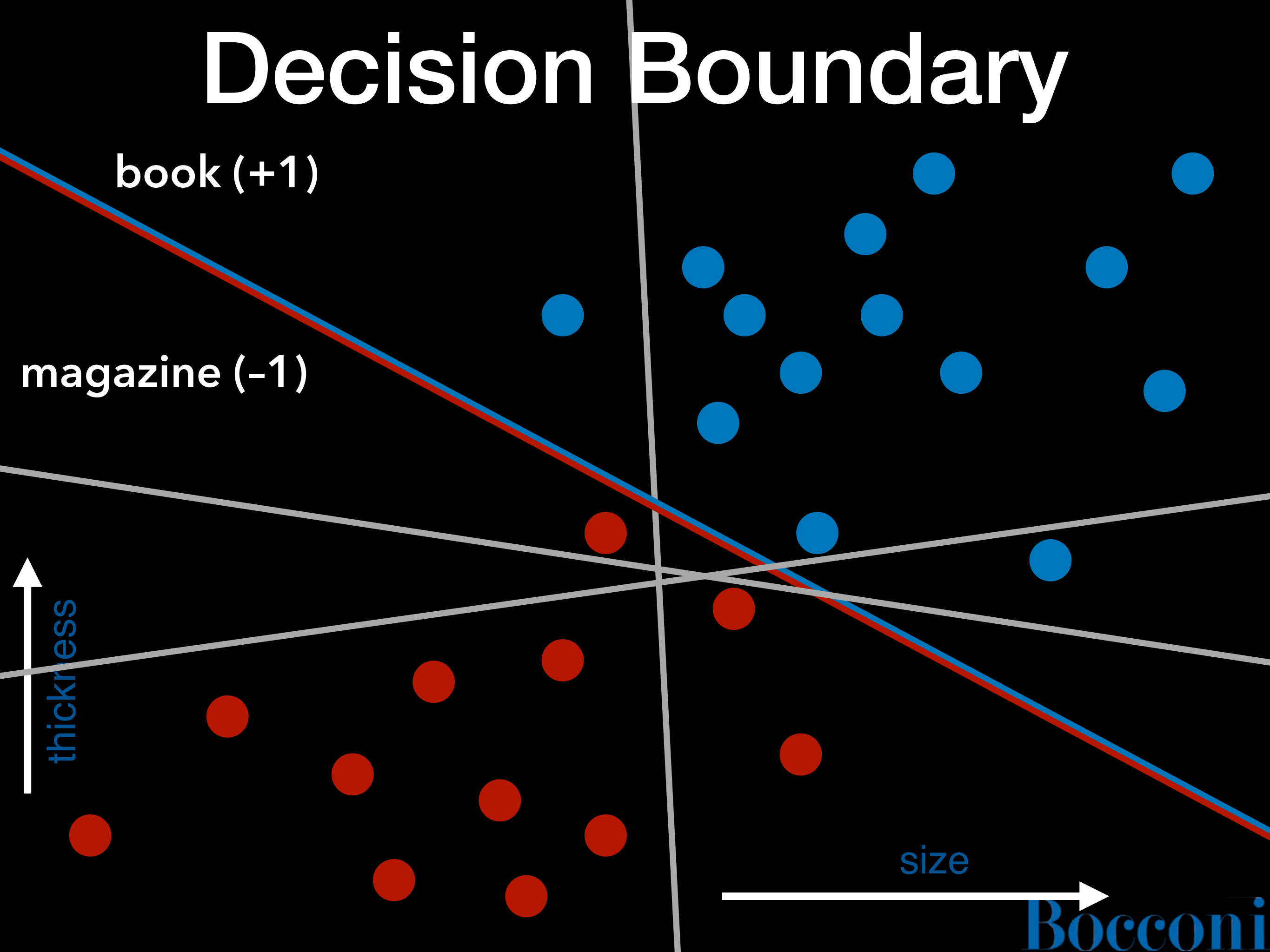
$$Y = a(WX + b)$$

Learning

Decision Boundary

book (+1)

magazine (-1)



thickness

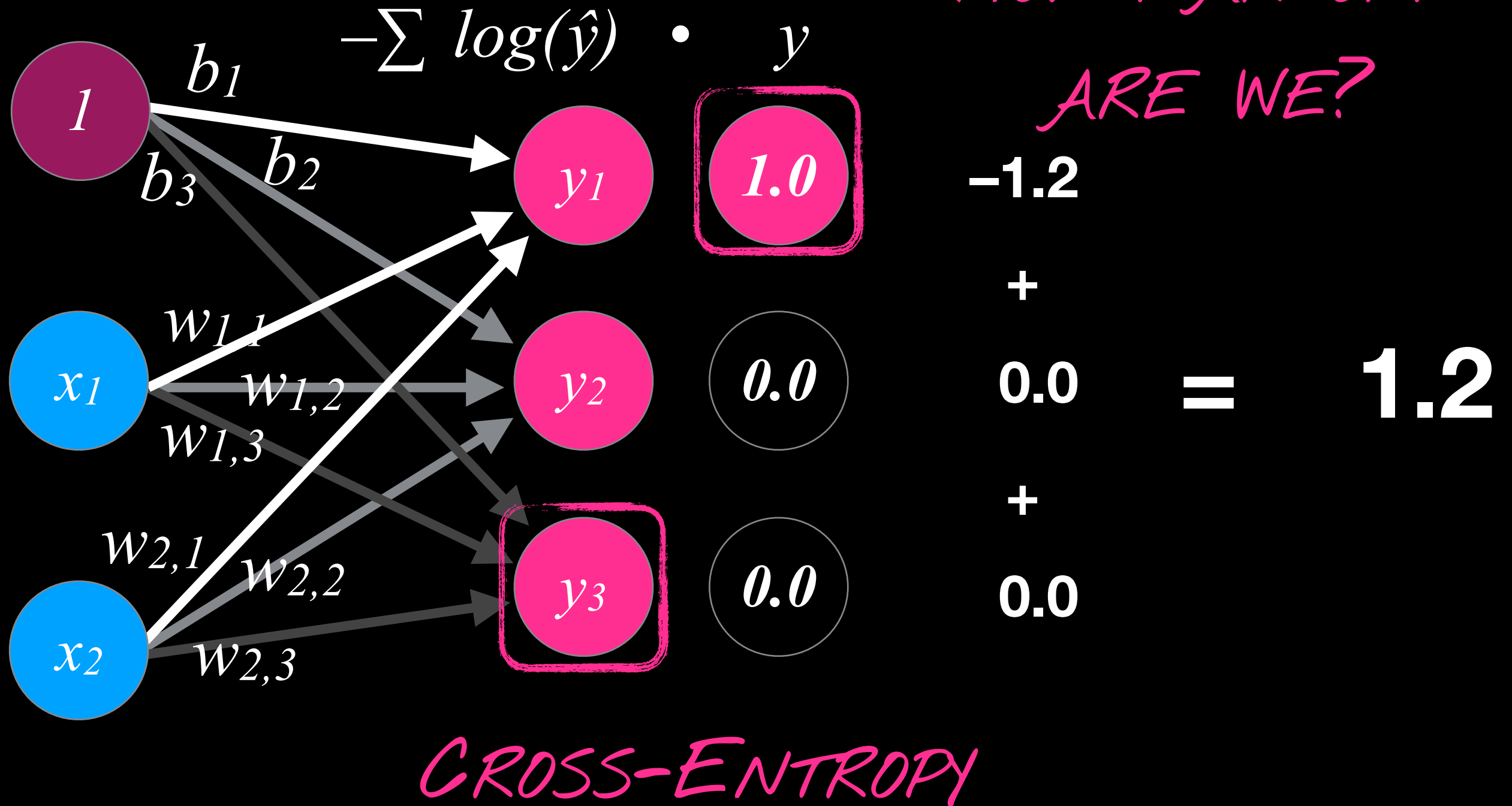
size

Bocconi

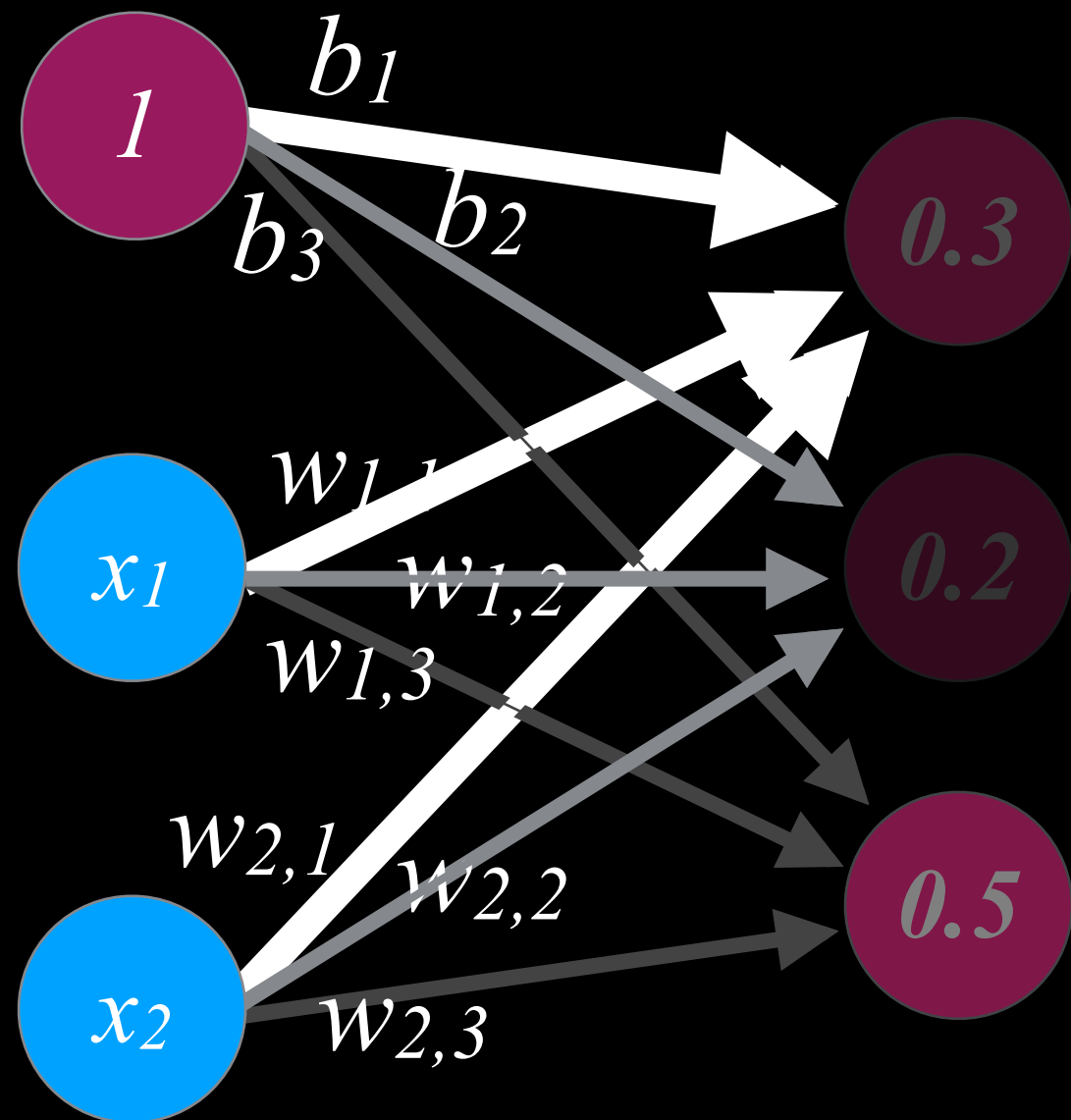
Error!

neutral positive

HOW FAR OFF
ARE WE?



Backpropagation

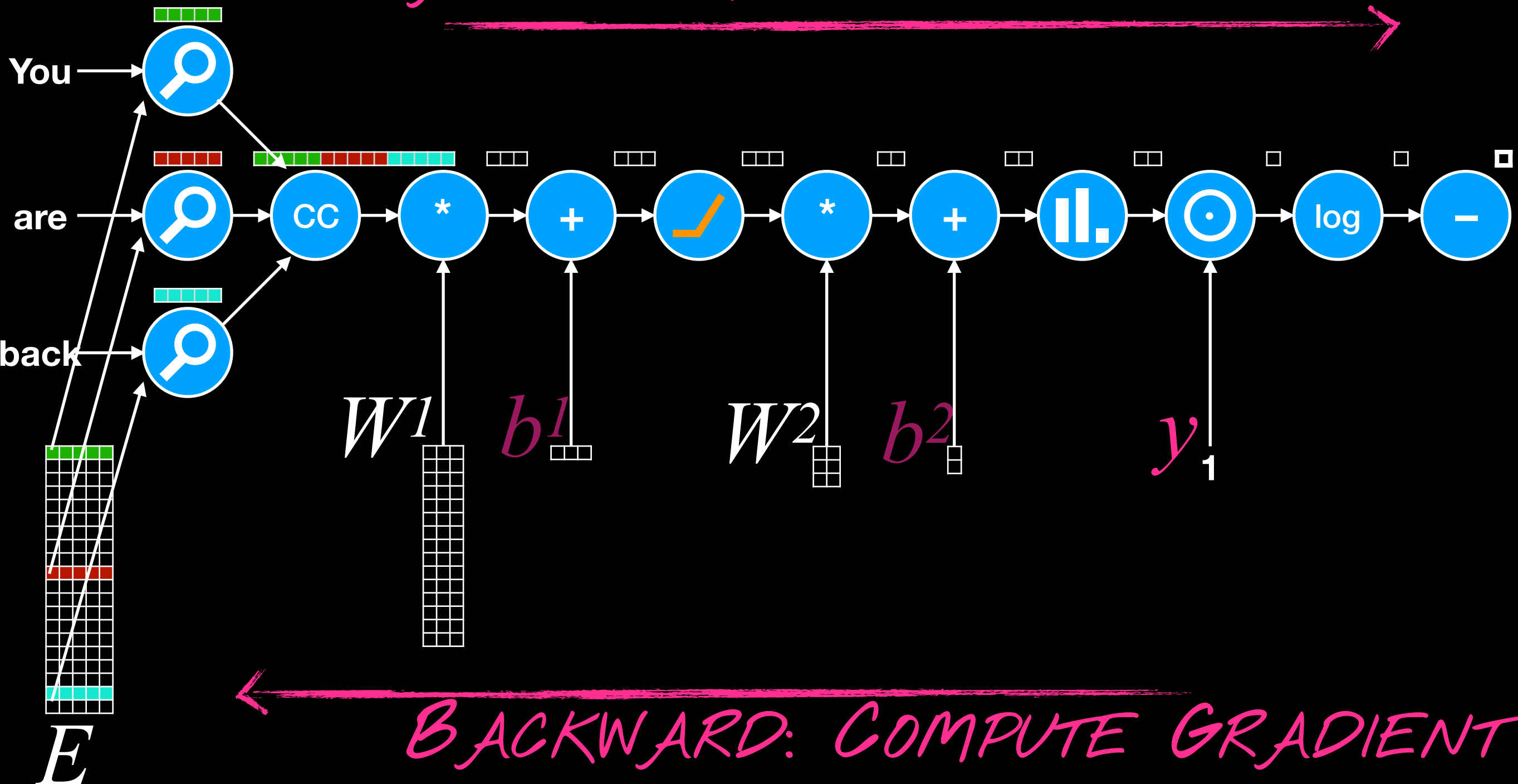


1.2

- Adjust weights/bias proportionately to change, using **Stochastic Gradient Descent**
- In deeper networks: compute effect on previous layer activation, then adjust *their* incoming weights accordingly, using **Chain Rule**
- If input layer is adjusted as well = learning representations

Computational Graph

FORWARD: COMPUTE ERROR



Regularization with Dropout

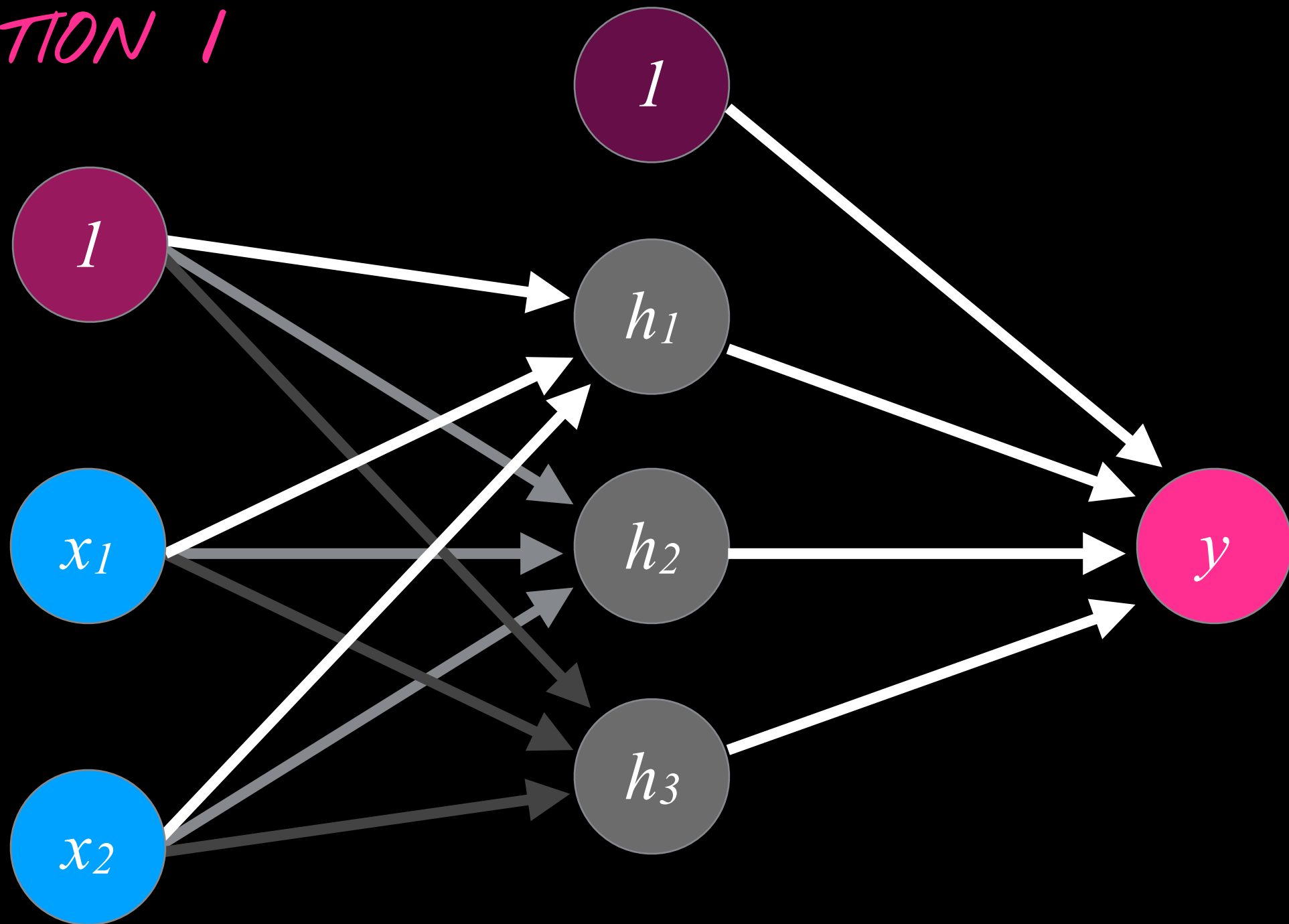
Overfitting



- ALVINN autonomous vehicle, trained on a particular stretch of road
- Drove off the road when turned around => focused on having a ditch on one side
- Idea: randomly remove nodes to avoid over-reliance

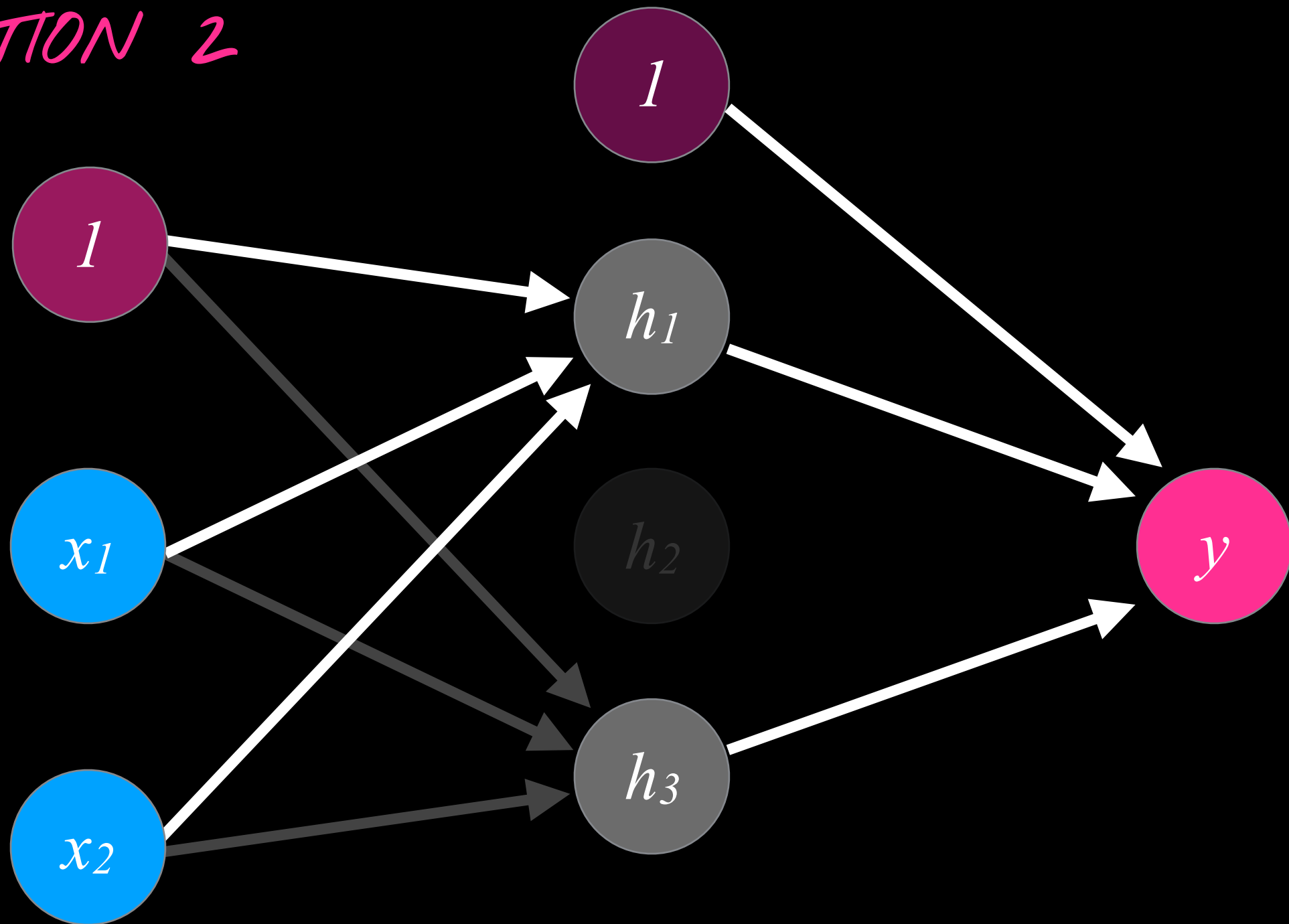
Dropout

ITERATION 1



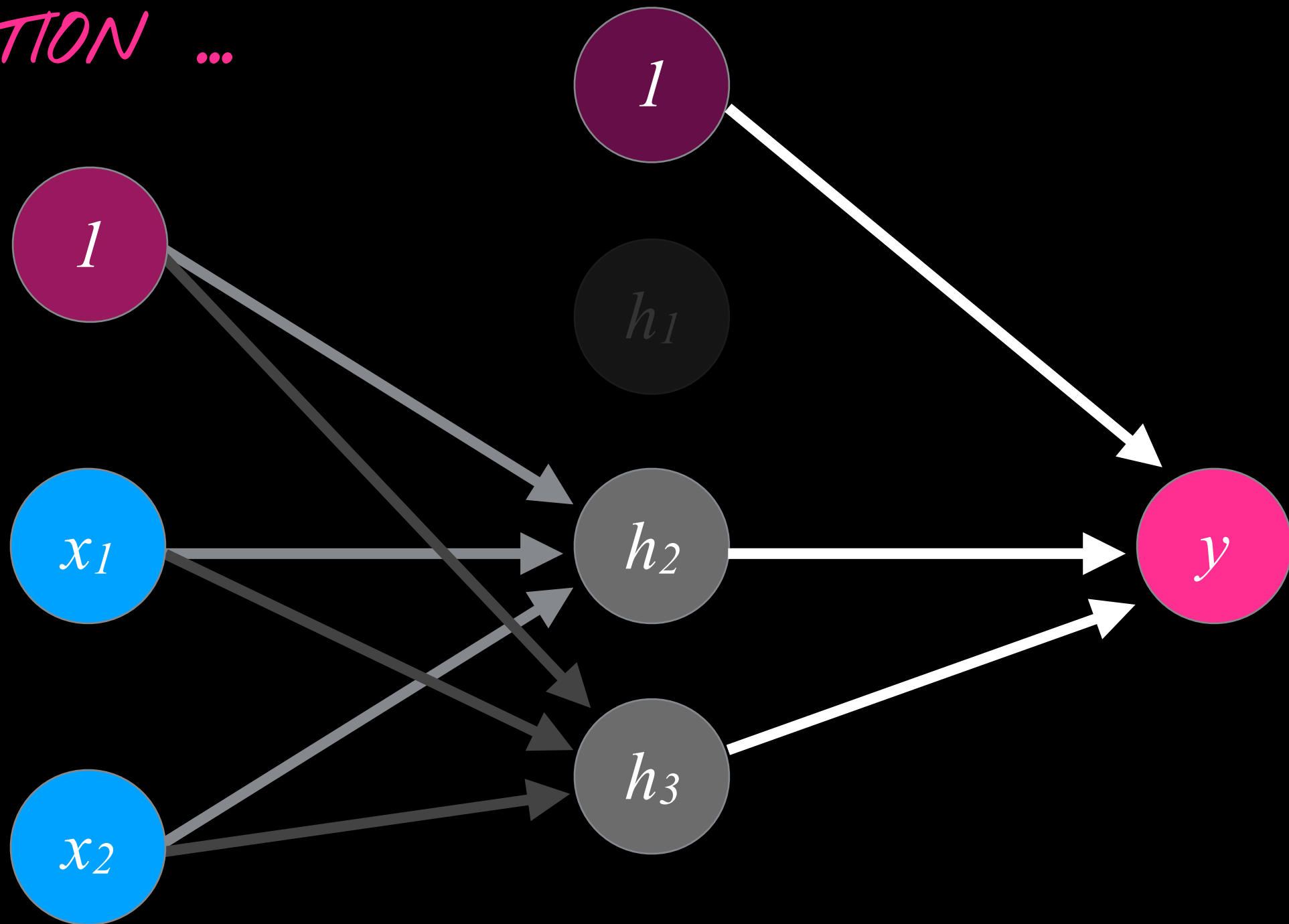
Dropout

ITERATION 2



Dropout

ITERATION ...



Wrapping up

Take Home Points

- The **perceptron** is the basic building block of NNs
- Several perceptrons are a **Multilayer Perceptron** or **Feedforward Network**
- Each layer is matrix multiplication wrapped in an **activation function** (usually **ReLU**)
- Training **backpropagates** an error through the network to change weights
- **Dropout** helps regularize networks by randomly deleting nodes

Moar Sources

- 3Blue1Brown: https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi
- Yoav Goldberg Primer: <https://arxiv.org/pdf/1510.00726.pdf>
- The Keras book
- ...