CS 333: NLP

Fall 2025

Prof. Carolyn Anderson
Wellesley College

Reminders

- HW 5 is due Thursday:
  - Regression with hand-crafted features to classify song lyrics by artist
- No class next Tuesday (Tanner)
- Quiz 7 next Friday (J&M Chapter 7)
- My next help hours: Thursday 4-5

AI AND HUMANITIES:

Human and Computational Language Models Can Help Each Other - Focusing on Linguistic Testing in Korean

Dr. Sanghoun Song, Associate Professor of Linguistics and Director of the Research Institute of Language and Information at Korea University

Wednesday, October 22

4:30 - 5:30 pm

Science Center - H401

Large Language Models: What can they tell us about language structure and acquisition

1-6pm on Nov. 6th
George Sherman Union, Boston University

https://web.sas.upenn.edu/societyforlanguagedevelopment/symposium

Stochastic Gradient Descent Recap

Learning in Supervised Classification

Supervised classification:

- We know the correct label y (either 0 or 1) for each x.
- But what the system produces is an estimate, $\hat{y}$

We want to set w and b to minimize the **distance** between our estimate $\hat{y}^{(i)}$ and the true $y^{(i)}$.

- We need a distance estimator: a **loss function** or a **cost function**
- We need an optimization algorithm to update w and b to minimize the loss.

Our goal: minimize the loss

For logistic regression, loss function is **convex**

- A convex function has just one minimum
- Gradient descent starting from any point is guaranteed to find the minimum
    - (Loss for neural networks is non-convex)

Gradients

The **gradient** of a function of many variables is a vector pointing in the direction of the greatest increase in a function.

Gradient Descent: Find the gradient of the loss function at the current point and move in the **opposite** direction.

Partial Derivative for Logistic Regression

Cross-Entropy Loss

$$L_{\mathrm{CE}}(\hat{y}, y) = -[y \log \sigma(w \cdot x + b) + (1 - y) \log (1 - \sigma(w \cdot x + b))]$$

Weight Update

$$w^{t+1} = w^t - \eta \frac{d}{dw} L(f(x; w), y)$$

Partial Derivative for Logistic Regression

Cross-Entropy Loss

$$L_{\mathrm{CE}}(\hat{y},y) = -[y\log\sigma(w\cdot x+b)+(1-y)\log(1-\sigma(w\cdot x+b))]$$

Weight Update

$$w^{t+1} = w^t - \eta\frac{d}{dw}L(f(x;w),y)$$

Derivative of Loss

$$\frac{d}{dw}L(f(x;w),y) = \frac{\partial}{\partial \mathbf{w}_j} - [y\log\sigma(\mathbf{w}\cdot\mathbf{x}+b)+(1-y)\log(1-\sigma(\mathbf{w}\cdot\mathbf{x}+b))]$$

Partial Derivative for Logistic Regression

Cross-Entropy Loss $L_{CE}(\hat{y},y) = -[y\log\sigma(w\cdot x+b)+(1-y)\log(1-\sigma(w\cdot x+b))]$

Weight Update $w^{t+1} = w^t - \eta\frac{d}{dw}L(f(x;w),y)$

Chain Rule $\frac{df}{dx} = \frac{du}{dv}\cdot\frac{dv}{dx}$

Derivative of σ(x) $\frac{d\sigma(z)}{dz} = \sigma(z)(1-\sigma(z))$

Derivative of ln(x) $\frac{d}{dx}\ln(x) = \frac{1}{x}$

Derivative of Loss $\frac{d}{dw}L(f(x;w),y) = \frac{\partial}{\partial \mathbf{w}_j} - [y\log\sigma(\mathbf{w}\cdot\mathbf{x}+b)+(1-y)\log(1-\sigma(\mathbf{w}\cdot\mathbf{x}+b))]$

$$= -\frac{\partial\, y\log\sigma(wx+b)}{\partial w_j} + \frac{\partial\,(1-y)\log[1-\sigma(wx+b)]}{\partial w_j}$$

$$= -\frac{y}{\sigma(wx+b)}\frac{\partial\sigma(wx+b)}{\partial w_j} - \frac{1-y}{1-\sigma(wx+b)}\cdot\frac{\partial\, 1-\sigma(wx+b)}{\partial w_j}$$

$$= \left[\frac{y}{\sigma(wx+b)} - \frac{1-y}{1-\sigma(wx+b)}\right]\frac{\partial\sigma(wx+b)}{\partial w_j}$$

Partial Derivative for Logistic Regression

Cross-Entropy Loss

$$L_{\text{CE}}(\hat{y}, y) = -[y \log \sigma(w \cdot x + b) + (1-y) \log (1 - \sigma(w \cdot x + b))]$$

Weight Update

$$w^{t+1} = w^t - \eta \frac{d}{dw} L(f(x;w), y)$$

Chain Rule

$$\frac{df}{dx} = \frac{du}{dv} \cdot \frac{dv}{dx}$$

Derivative of σ(x)

$$\frac{d\sigma(z)}{dz} = \sigma(z)(1 - \sigma(z))$$

Derivative of ln(x)

$$\frac{d}{dx} \ln(x) = \frac{1}{x}$$

Derivative of Loss

$$\begin{aligned}
\frac{d}{dw} L(f(x;w), y) &= \frac{\partial}{\partial \mathbf{w}_j} - [y \log \sigma(\mathbf{w} \cdot \mathbf{x} + b) + (1-y) \log (1 - \sigma(\mathbf{w} \cdot \mathbf{x} + b))] \\
&= -\left[\frac{\partial}{\partial \mathbf{w}_j} y \log \sigma(\mathbf{w} \cdot \mathbf{x} + b) + \frac{\partial}{\partial \mathbf{w}_j} (1-y) \log [1 - \sigma(\mathbf{w} \cdot \mathbf{x} + b)] \right] \\
&= -\frac{y}{\sigma(\mathbf{w} \cdot \mathbf{x} + b)} \frac{\partial}{\partial \mathbf{w}_j} \sigma(\mathbf{w} \cdot \mathbf{x} + b) - \frac{1-y}{1 - \sigma(\mathbf{w} \cdot \mathbf{x} + b)} \frac{\partial}{\partial \mathbf{w}_j} 1 - \sigma(\mathbf{w} \cdot \mathbf{x} + b) \\
&= -\left[\frac{y}{\sigma(\mathbf{w} \cdot \mathbf{x} + b)} - \frac{1-y}{1 - \sigma(\mathbf{w} \cdot \mathbf{x} + b)} \right] \frac{\partial}{\partial \mathbf{w}_j} \sigma(\mathbf{w} \cdot \mathbf{x} + b) \\
&= -\left[\frac{y - \sigma(\mathbf{w} \cdot \mathbf{x} + b)}{\sigma(\mathbf{w} \cdot \mathbf{x} + b)[1 - \sigma(\mathbf{w} \cdot \mathbf{x} + b)]} \right] \sigma(\mathbf{w} \cdot \mathbf{x} + b)[1 - \sigma(\mathbf{w} \cdot \mathbf{x} + b)] \frac{\partial(\mathbf{w} \cdot \mathbf{x} + b)}{\partial \mathbf{w}_j} \\
&= [\sigma(\mathbf{w} \cdot \mathbf{x} + b) - y] \mathbf{x}_j
\end{aligned}$$

Deriving cross-entropy loss for multi-label classification

Goal: maximize probability of the correct label $p(y|x)$

$$
\begin{aligned}
L_{\mathrm{CE}}(\hat{\mathbf{y}},\mathbf{y}) &= -\sum_{k=1}^{K} \mathbf{y}_k \log \hat{\mathbf{y}}_k \\
&= -\log \hat{p}(\mathbf{y}_c = 1|\mathbf{x}) \quad \text{(where } c \text{ is the correct class)} \\
&= -\log \frac{\exp(\mathbf{w_c}\cdot\mathbf{x}+b_c)}{\sum_{j=1}^{K}\exp(\mathbf{w_j}\cdot\mathbf{x}+b_j)} \quad (c \text{ is the correct class})
\end{aligned}
$$

The true probabilities of all but one class will be zero.

How much do we move in that direction ?

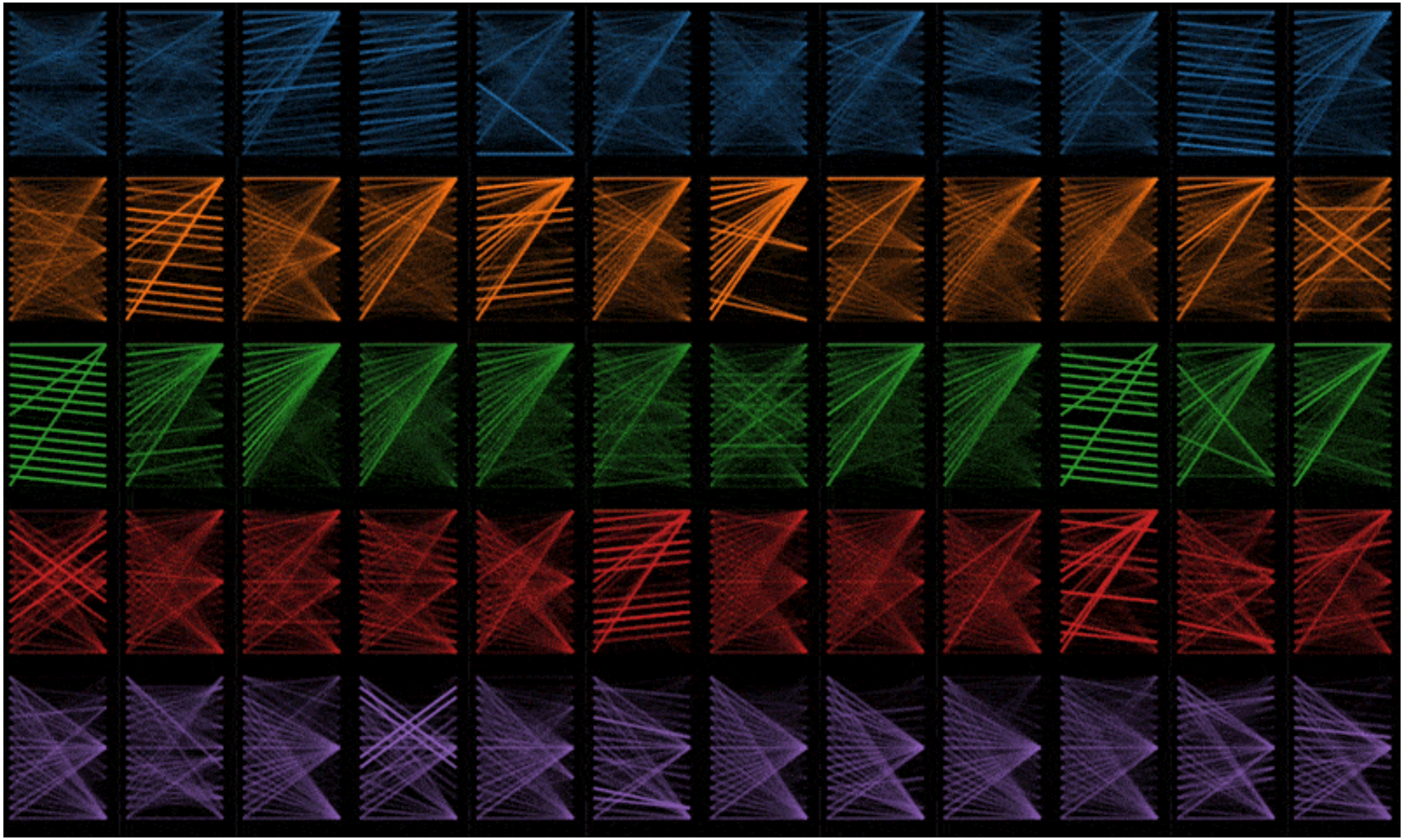
- The value of the gradient (slope in our example) $\frac{d}{dw}L(f(x;w),y)$ weighted by a **learning rate** η
- Higher learning rate means that we make bigger adjustments to the weights

$$w^{t+1} = w^t - \eta \frac{d}{dw}L(f(x;w),y)$$

www.i-am.ai/gradient-descent.html

Neural Networks

This is not your brain



https://github.com/jessevig/bertviz

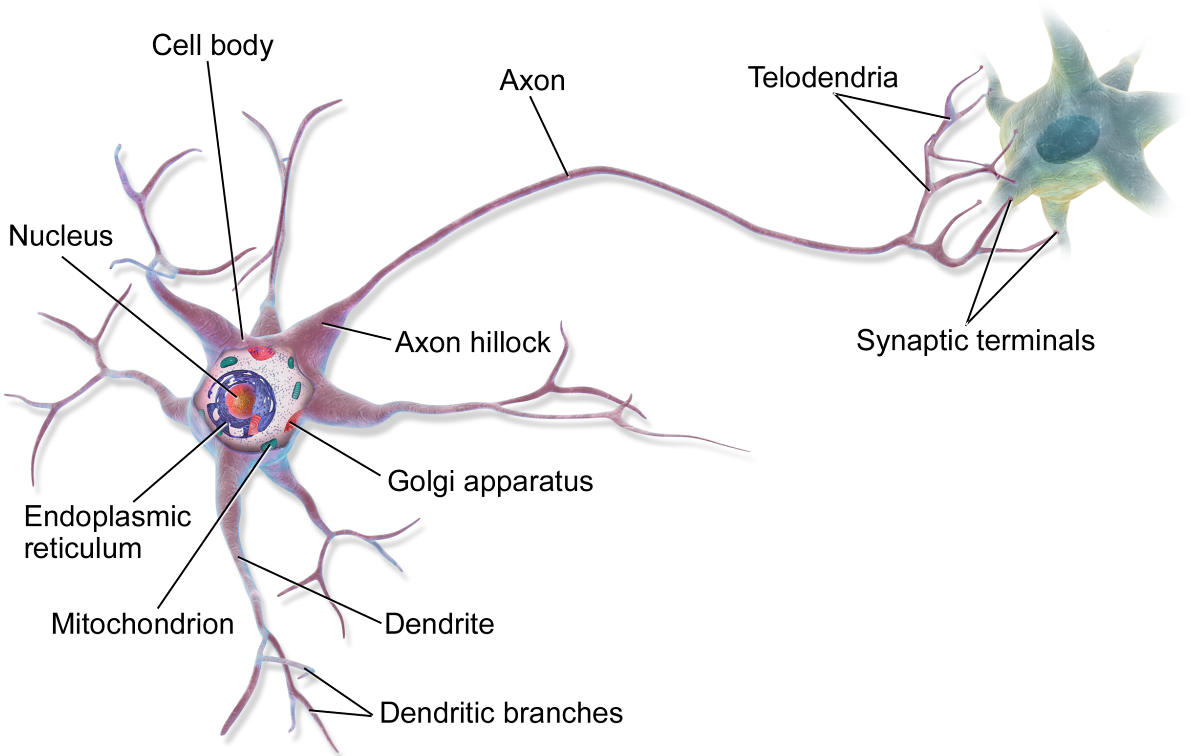
It's a large language model (neural network)

Most contemporary models have billions of parameters (GPT3: 175 billion); some may have trillions

This is someone's brain

estimated to have 100 trillion synapses connecting 100 billion neurons

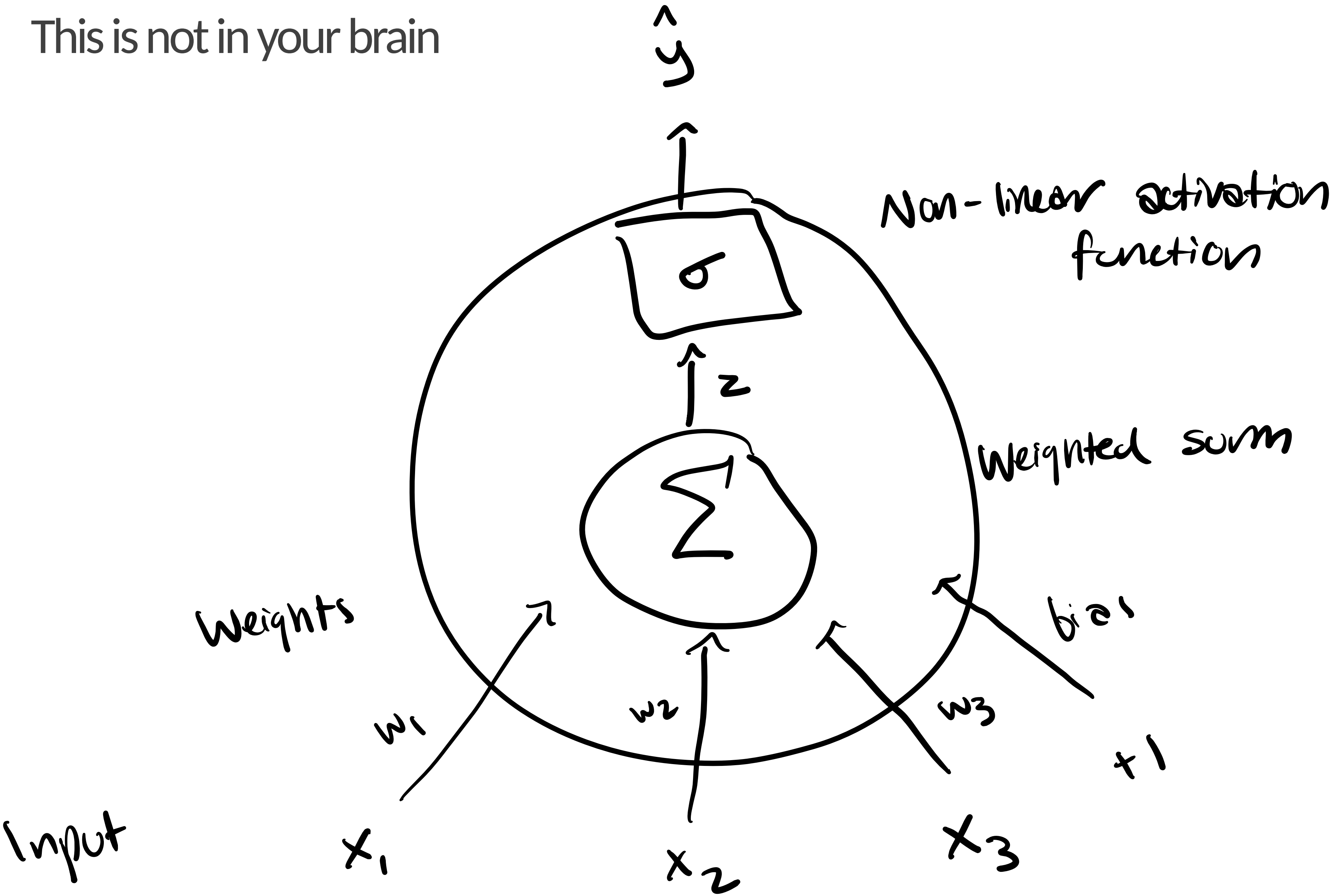
This is in your brain



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

Neural Network Unit

This is not in your brain



Units in Neural Networks

Neural unit

Takes a weighted sum of inputs
(plus bias)

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

$$= w \cdot x + b$$

$$\hat{y} = a = f(z)$$

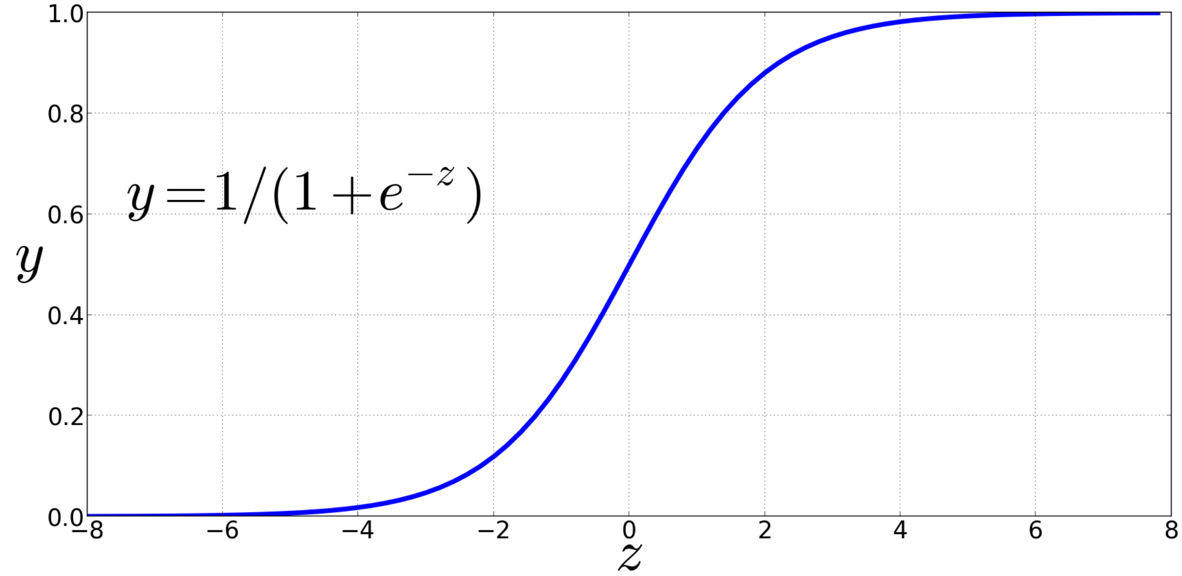
(could be sigmoid)

Non-Linear Activation Functions

We're already seen the sigmoid for logistic regression:

Sigmoid

$$\sigma(z) = \frac{1}{1+e^{-z}}$$



Final function of a unit:

Regression:

$$\hat{y} = \sigma(wx + b)$$

Neural Network Unit:

$$\hat{y} = f(wx + b)$$

where f is some activation function

Spot the differences

Neural Network Unit

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

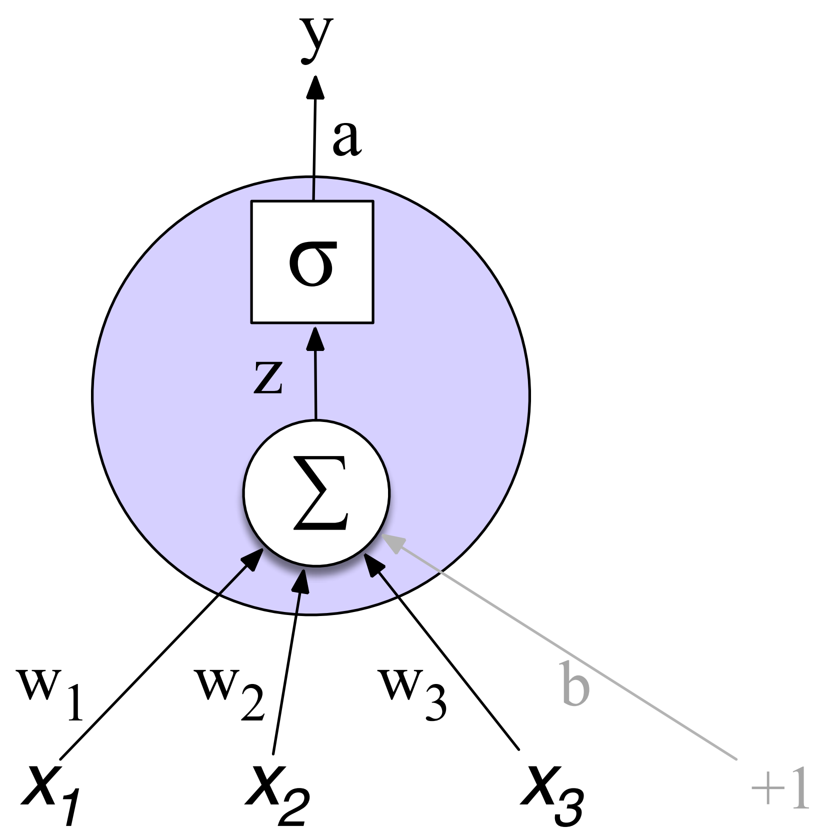
$$\hat{y} = f(w \cdot x + b)$$

Logistic Regression

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

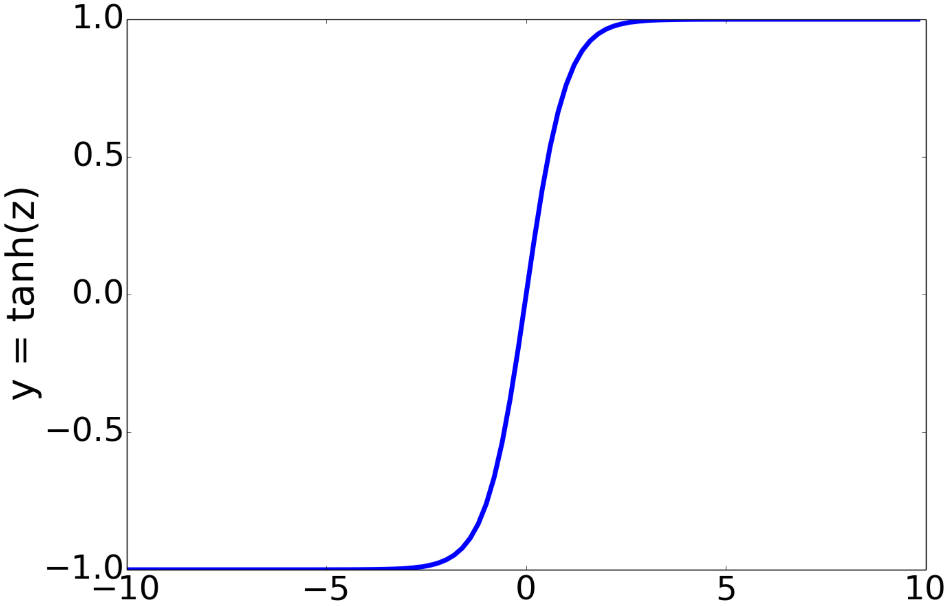
$$p(y=1 \mid x) = \sigma(w \cdot x + b)$$

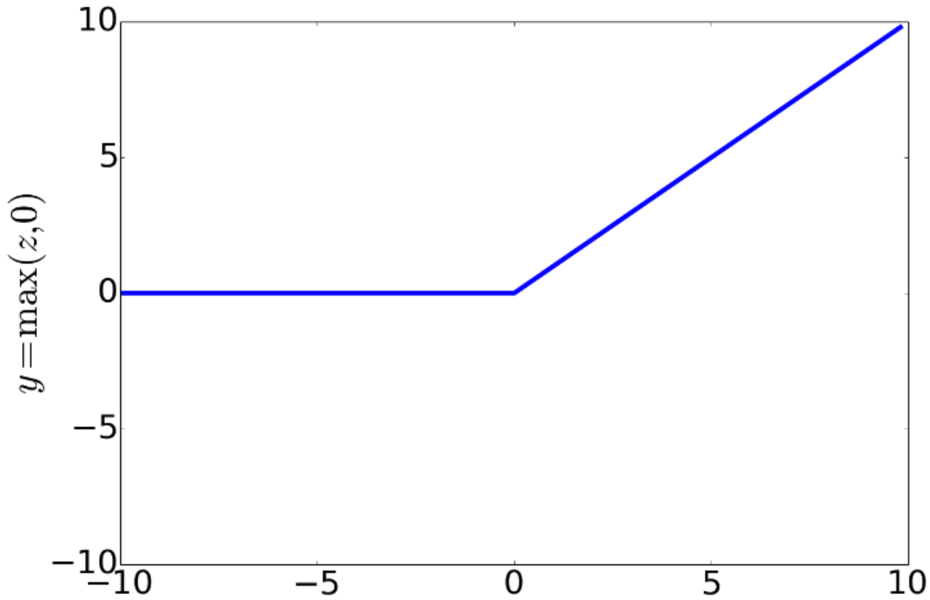
Final unit again



Non-Linear Activation Functions besides sigmoid

Most Common:





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 neural unit

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

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

Solving AND

Slides borrowed from Jurafsky & Martin Edition 3

Deriving AND

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

Goal: return 1 if x1 and x2 are 1

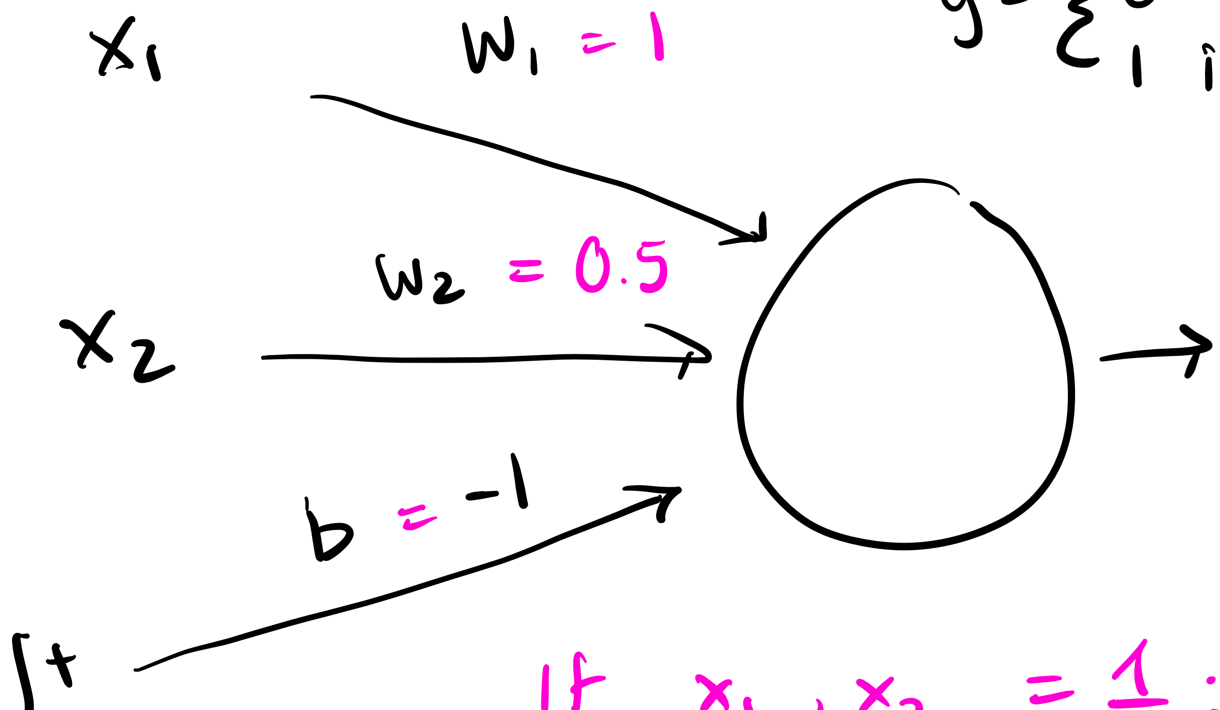
AND

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

Deriving AND

Goal: return 1 if x1 and x2 are 1

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



If $x_1 = 1$ and $x_2 = 0$:

$z = 1 + 0 - 1 = 0$

$y =$

If $x_1, x_2 = \underline{1}$:

$z = 1 + 0.5 - 1 = 0.5$

$y = 1$

Exercise: solving OR

OR

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

Deriving OR

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

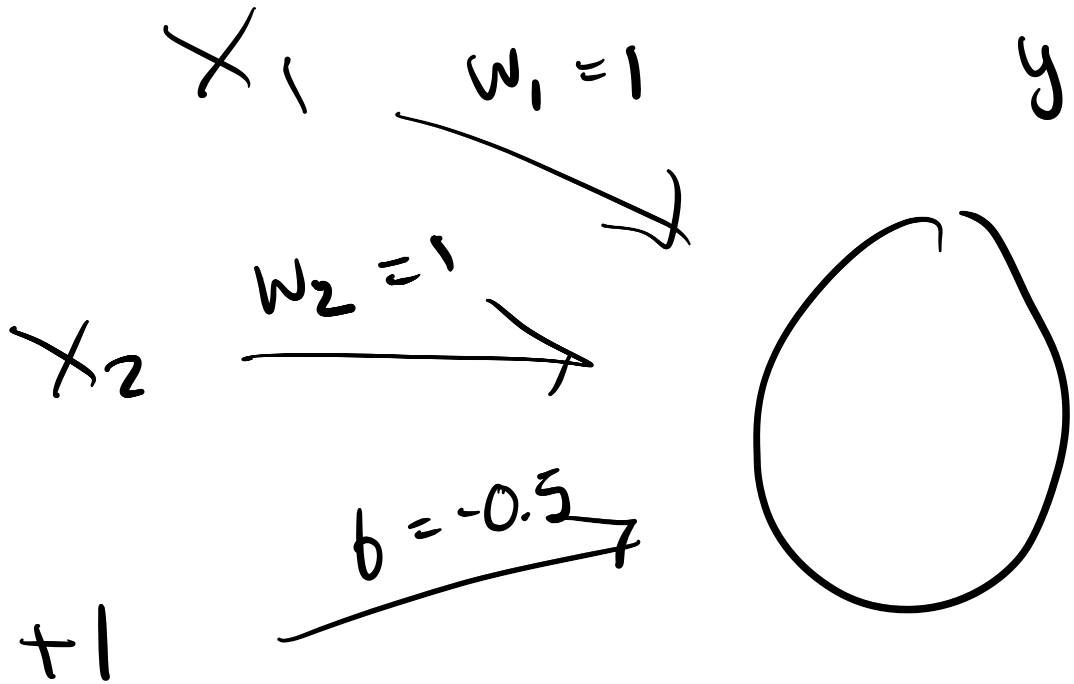
Goal: return 1 if either input is 1

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



if $x_1 = 0$, $x_2 = 0$:

$z = 1 \cdot 0 + 1 \cdot 0 + -0.5 =$

-0.5

$y = 0$

if $x_1 = 1$, $x_2 = 0$:

$z = 1 \cdot 1 + 0 \cdot 1 + -0.5 = 0.5$

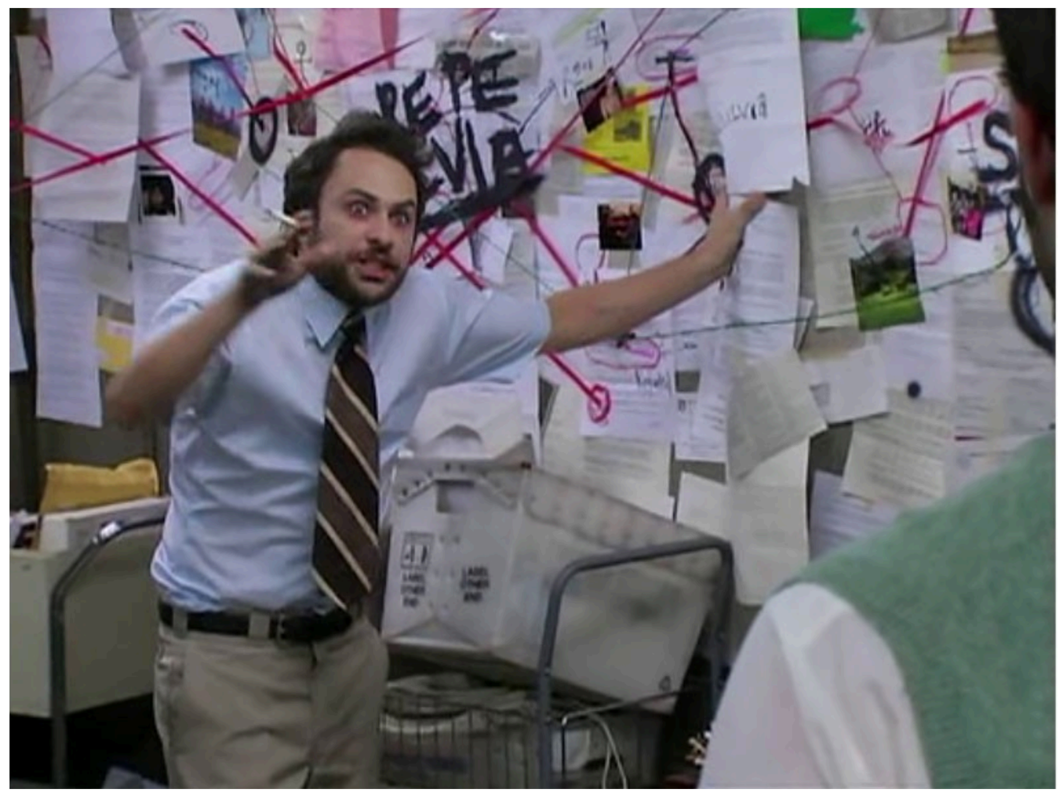
$y = 1$

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

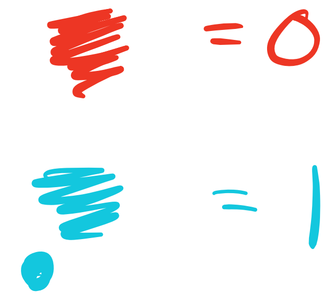
$w_1 x_1 + w_2 x_2 + b = 0$

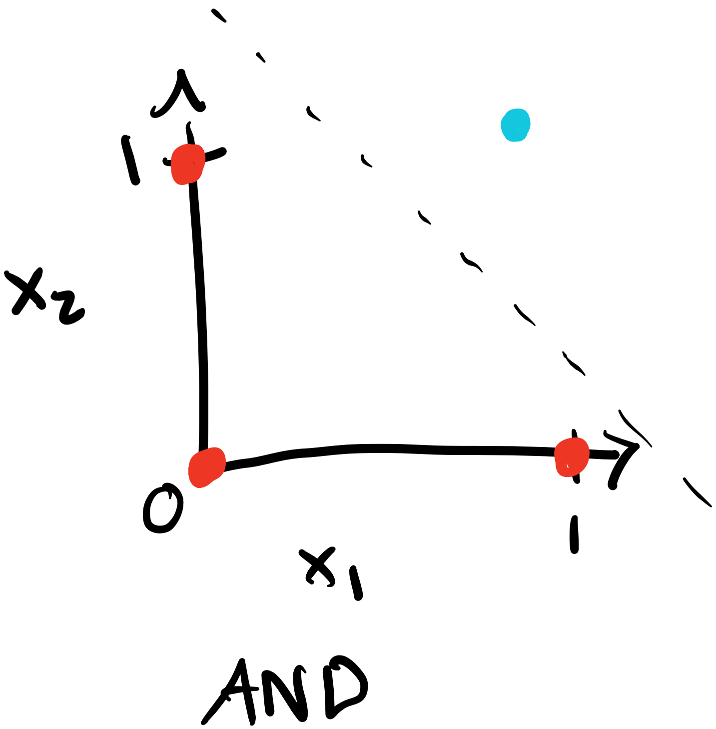
(in standard linear format: $x_2 = (-w_1/w_2)x_1 + (-b/w_2)$)

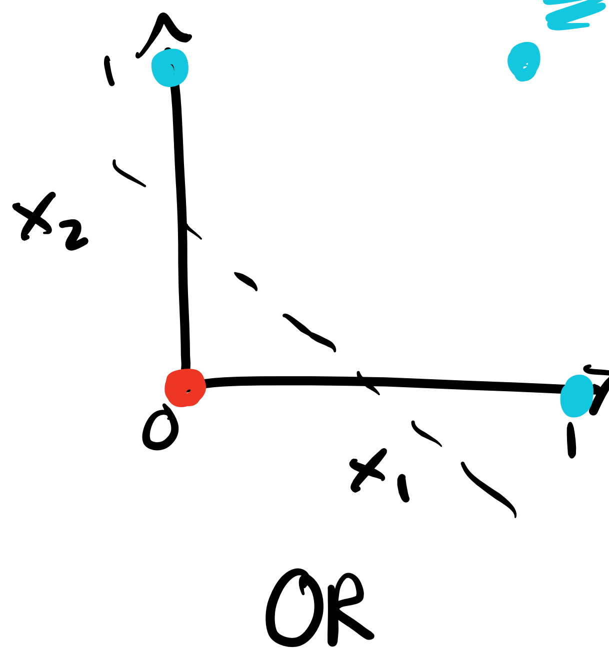
This line acts as a **decision boundary**

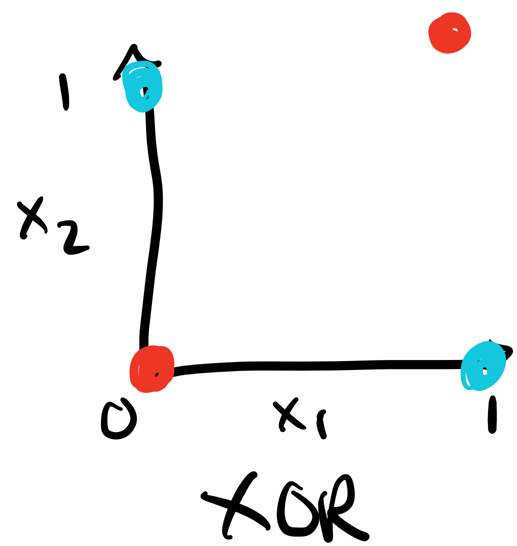
- 0 if input is on one side of the line
- 1 if on the other side of the line

Decision boundaries









XOR is not linearly separable.

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