

# Lecture 21

04/05/2022

⇒ Before we get to neural networks, we'll need some tools (in order to train the networks).  
==

- Loss Functions ~ or "How do I penalize model errors?"

--- lots of choices ⇒  $y = \text{label (data)}$   
 $f(x) = \text{model prediction}$

$$\begin{aligned} * \text{ L2 loss} &= [y - f(x)]^2 \\ * \text{ L1 loss} &= |y - f(x)| \quad (\text{Huber}) \end{aligned}$$

⇒ the **losses** are aggregated over all data in order to make the **COST FUNCTION**

| LOSS | COST             |
|------|------------------|
| L2   | $\text{MSE}/x^2$ |
| L1   | MAE              |

⇒  $\text{MSE}/x^2$  is the most popular cost function  
⇒ helps for it to be easily **differentiable**.  
⇒ Often attempt several cost functions when tuning a ML / NN architecture.

## Gradient Descent



\* What's the best way to find the valley floor?  
 ↳ start walking downwards ...  
 i.e. move AGAINST THE GRADIENT

$$\Delta y = \frac{dy}{dx} \Delta x$$

$$y^{i+1} = y^i - \frac{dy}{dx} \Delta x$$

⇒ more generally ...  $MSE(\theta) = \frac{1}{N} (Y - X\theta)^T (Y - X\theta)$

$$\nabla_{\theta} MSE(\theta) = \frac{2}{N} X^T (X\theta - Y)$$

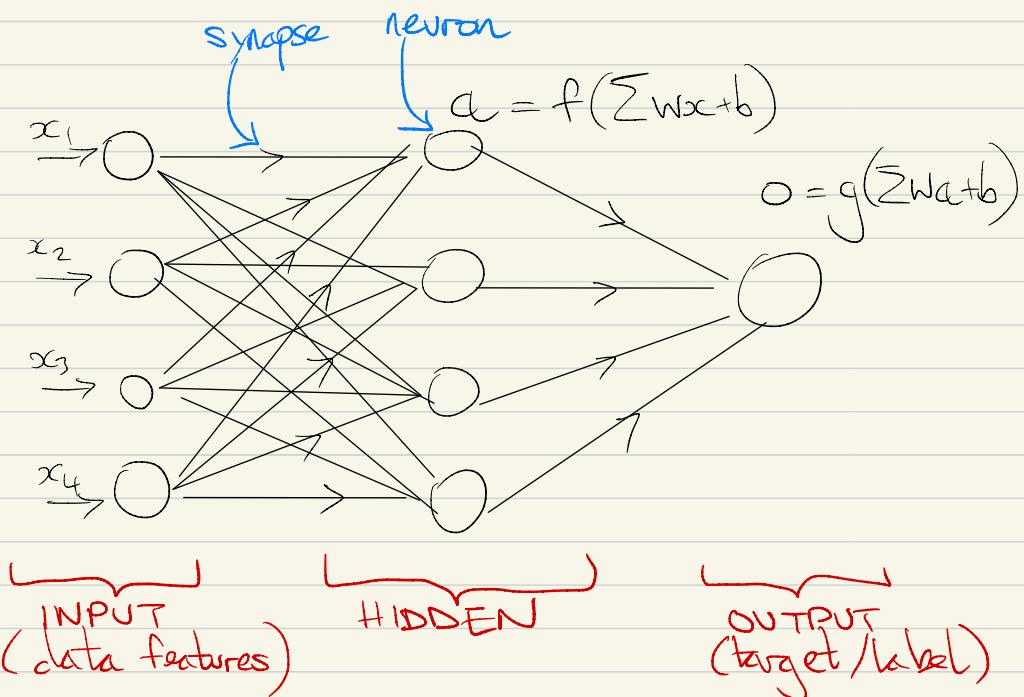
$$\Rightarrow \theta^{next} = \theta^{now} - \eta \nabla_{\theta} MSE(\theta)$$

learning rate  
---not too big/small

- Ada Boost ~ Adaptive Boosting

↳ Meta-estimator that uses gradient descent to tune classification performance.

- Neural Networks ~ ANNs are tools based on a metaphor with neurons.

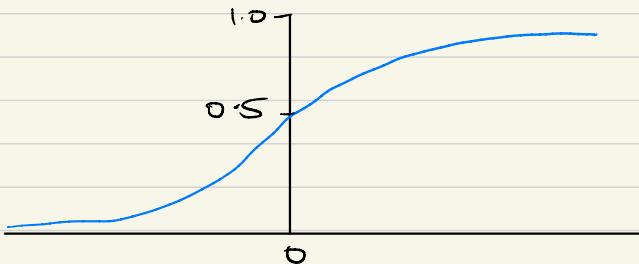


\* Synapses in hidden layers form a weighted sum of all inputs, add a bias, then pass to a neuron,

$$\sum_i w_i x_i + b$$

- \* Neurons sum this input, then pass it through a non-linear activation function

e.g. SIGMOID ACTIVATION FUNCTION



$$a = \frac{1}{1 + e^{-z}}$$

\* Backpropagation = clever way of finding gradient of cost function analytically.

\* How many layers?  $\Rightarrow$  more layers gives more non-linearity.

\* How many neurons?  $\Rightarrow$  roughly between  $\sim 2\sqrt{N_{\text{input}} \cdot N_{\text{output}}}$

\* Which activation?

$\begin{cases} \text{sigmoid} = \text{binary classify} \\ \text{ReLU} = \text{hidden layers} \\ \text{softmax} = > 2 \text{ classes classification} \end{cases}$

\* Regularization can also help constrain the network architecture optimization.