Activation Functions (and gradient vanishing)

- as seen in the NN notes, an activation function in a NN determines wether a neuron should be activated (ie pass info forward) it decides how much signal a neuron passes forward to the next layer. Without it a NN would become on giant linear equation and become useless. Activation function introduce non-linearity to the network allowing it to learn.

Ex: after a layer computes $z = Wx + b$ the activation function transforms it into something non-linear like Relu(z) so the next layer can learn useful patterns (in short if a result is over some threshold the neuron fires and passes information in form of a signal it on/off and how much is it "in" by, to the next layer)

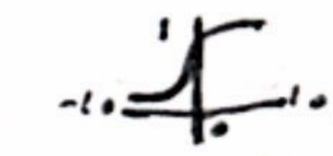
Numeric Ex: input: $x = [0.5, 0.3]$ weights: $w = [0.4, 0.6]$ bias: $b = 0.1$

weighted sum of Neuron: $z = Wx + b = 0.5 \times 0.4 + 0.3 \times 0.6 + 0.1 = 0.32$

apply activation func of choice ReLU$(0.32) = \max(0, \underset{z}{0.32}) = \underline{0.32}$

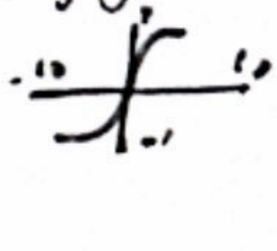
Types of activation functions (common ones)

- ReLU: $\max(0, x)$ ⇒ Simple fast, and works well in practice, but can kill neurons when inputs go negative for too long

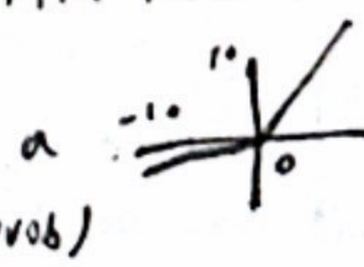


- Sigmoid: squashes values to $[0, 1]$ good for probabilities, bad for deep layers as it makes gradients vanish meaning ⇒ During backprop (method used in NN to learn from mistakes by calculating how much each weight contributed to the error, then adjusting weights in direction that reduces error more) from output layer to input/ the gradient you need to update early layers shrinks so much it basically becomes 0 When that happens layers stop learning they freeze while later layers do the learning. Because some functions squash values into tiny ranges where the func's derivatives are close to zero, multiply enough near zero derivatives across many layers and signal dies.

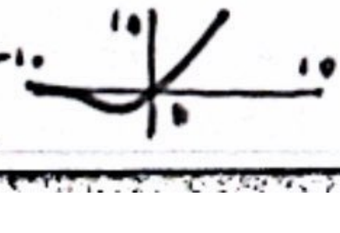
- Tanh: like sigmoid but centered at 0; $\tanh(0) = 0$, still suffers from vanishing grad



- Leaky Relu: $\max(0.1x, x)$ ⇒ fixes dead neurons by allowing a small slope for negatives (a band-aid for ReLUs dying neuron prob)



- GeLU: used in transformers, smooth, non linear, better gradient flow than ReLU works better since it reflects the idea that small neg values should not be treated as strictly dead



sigmoid gradient is the vector that points in NOTE: direction needed to minimize error