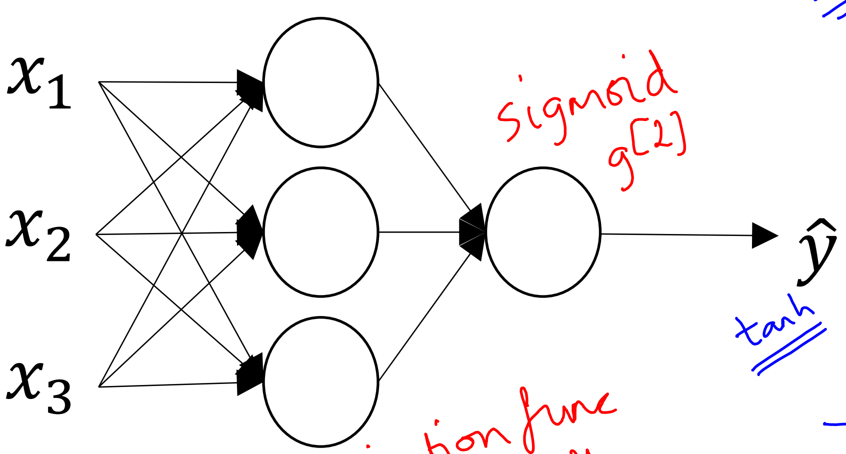
deeplearning.ai

One hidden layer Neural Network

Activation functions

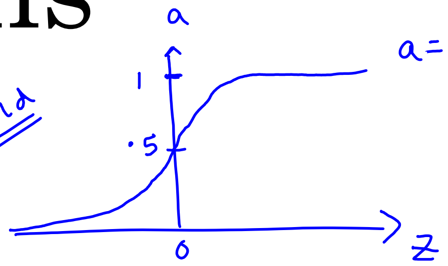
Activation functions $\rightarrow g$

$\tanh = g^{[1]}$



Activation func can be diff for diff layers

sigmoid

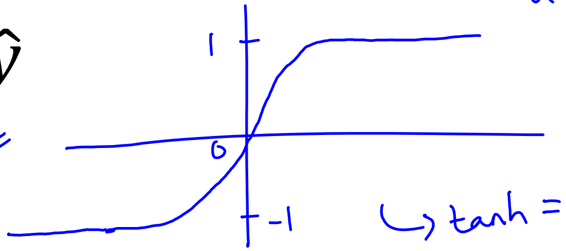


$a = \frac{1}{1+e^{-z}}$

Never use sigmoid, except maybe in the o/p layer

eg In a classification problem If you want o/p either 0 or 1 then sigmoid gives that to you, tanh may give -ve values

tanh



$a = \tanh(z)$

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

↳ tanh = shifted version of sigmoid

tanh is better than sigmoid

why? → Since tanh shifts mean of activation values from .5 to 0 ∴ its better, why?

↳ It has a similar effect as if you were standardizing your data

it centers the data to the next layer around 0 rather than .5

Given x:

$$z^{[1]} = W^{[1]}x + b^{[1]}$$

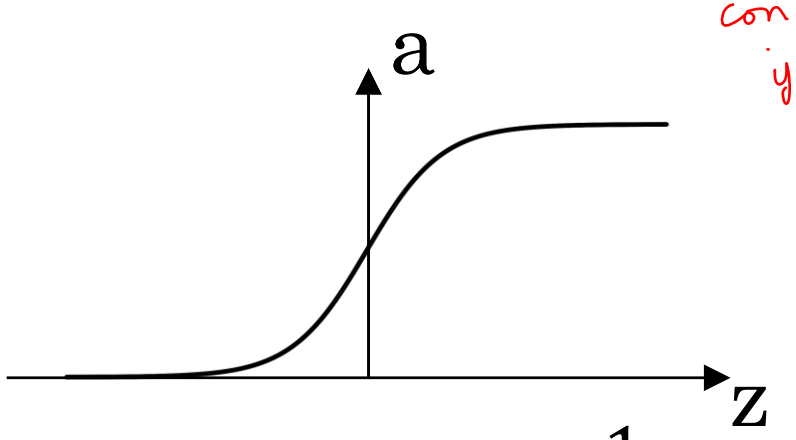
$$a^{[1]} = \sigma(z^{[1]}) \rightarrow \text{Activation func (doesnt have to be sigmoid)}$$

↳ can be Any $g(z)$

$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$

$$a^{[2]} = \sigma(z^{[2]})$$

Pros and cons of activation functions



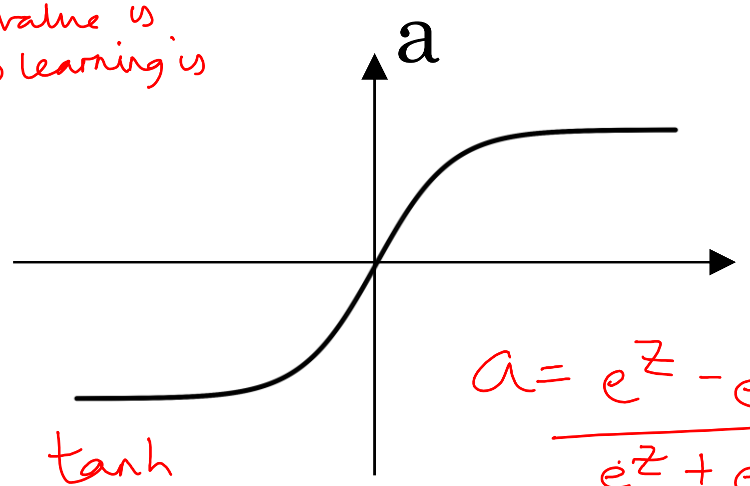
sigmoid: $a = \frac{1}{1 + e^{-z}}$

con of both sigmoid + tanh is that if z is large or small, slope value is small ⇒ gradient is small ⇒ learning is slow

Rules

① Binary classification ⇒ sigmoid

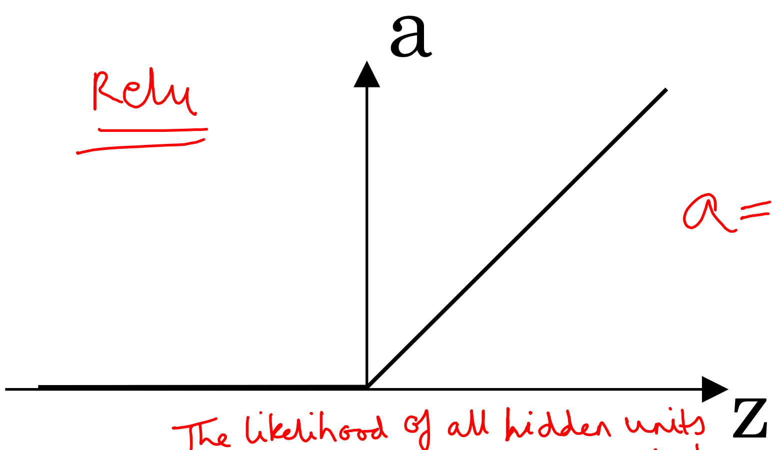
② otherwise use Relu



tanh

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

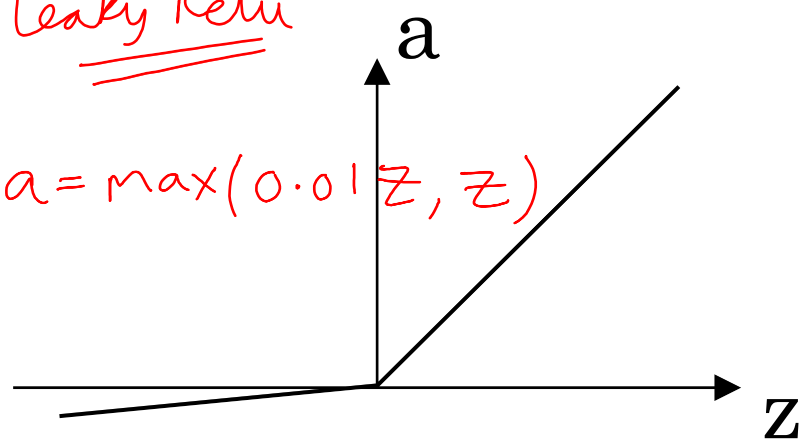
Relu



$a = \max(0, z)$

The likelihood of all hidden units being ≤0, ie, all "z" are 0 in the NN, or all "z" are <0 is very small ⇒ learning will still happen even if some units are >0

Leaky Relu



$a = \max(0.01z, z)$