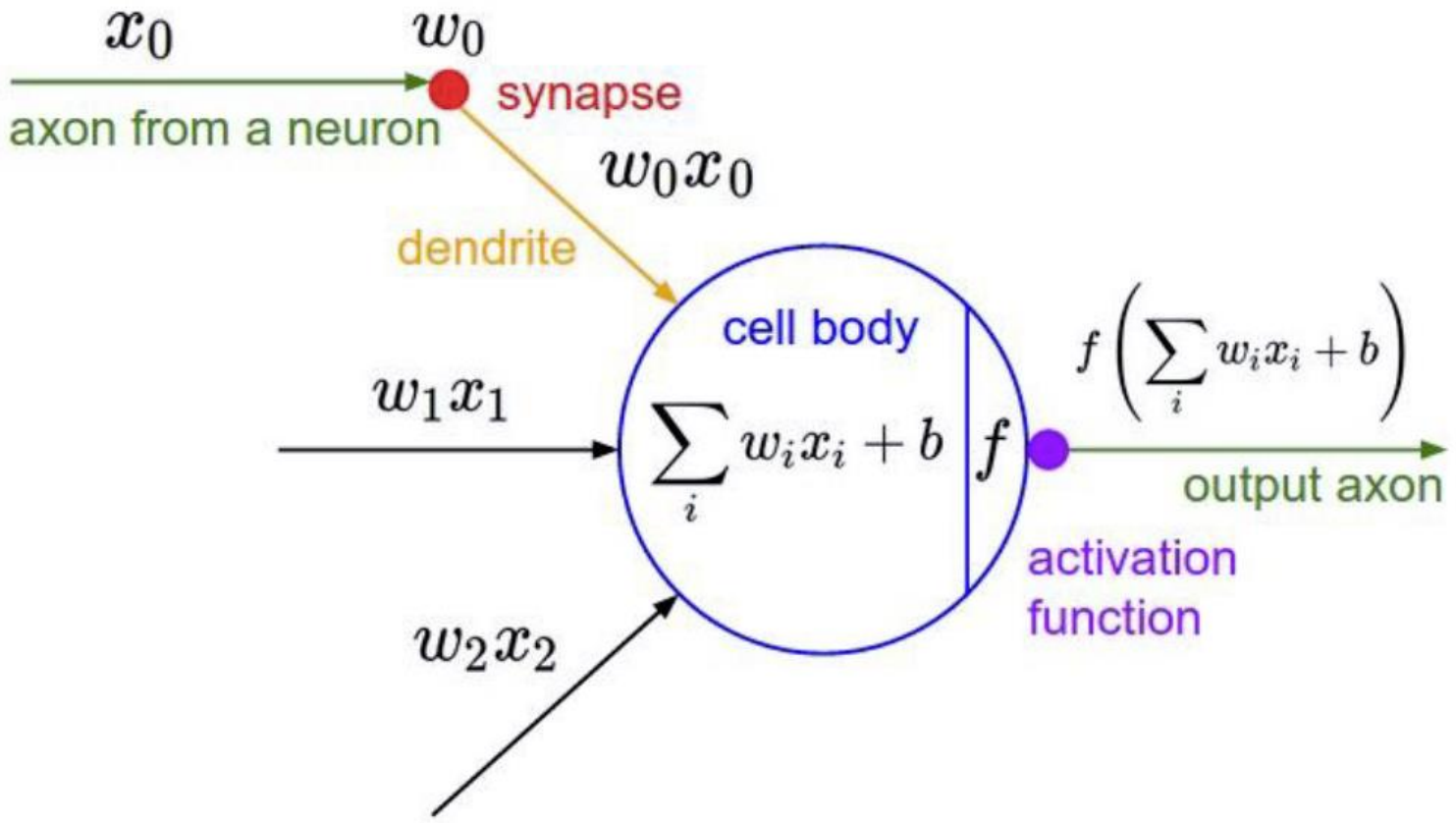
Shallow Neural Networks

Dai Bui

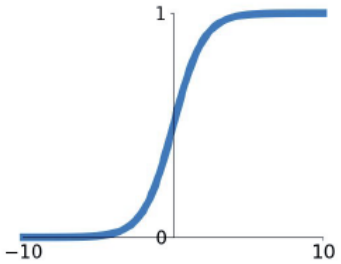
Activation Functions



Activation Functions

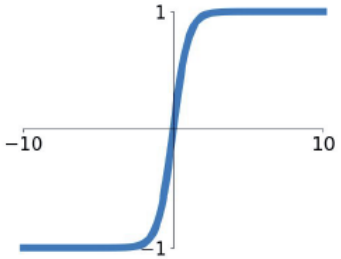
Sigmoid

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



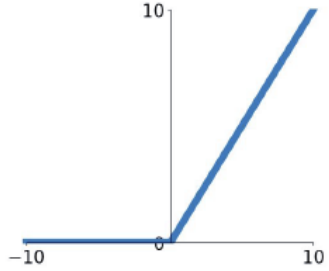
tanh

$$\tanh(x)$$



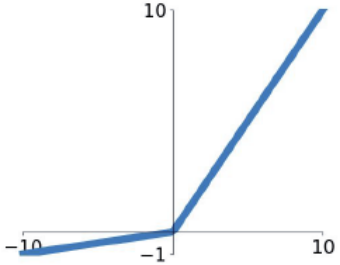
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

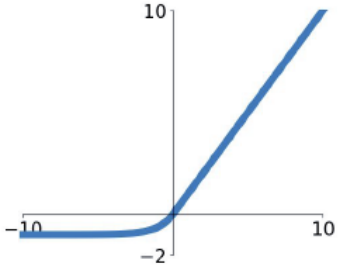


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

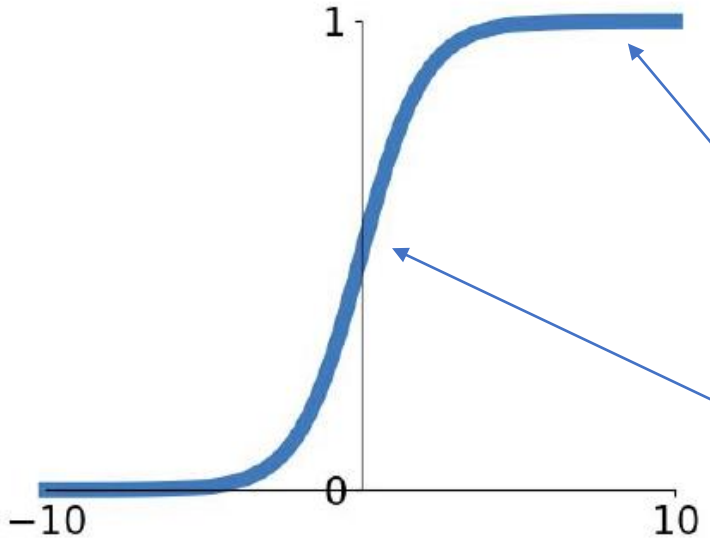
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Activation Functions: Sigmoid

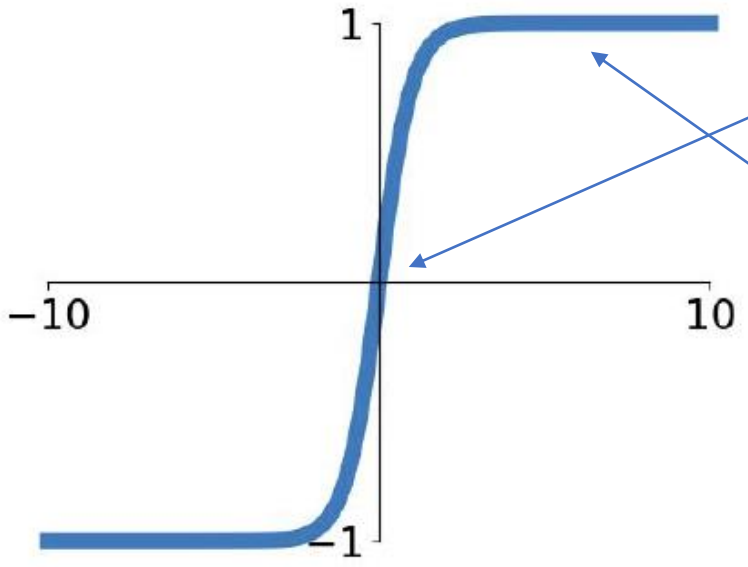
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



Sigmoid

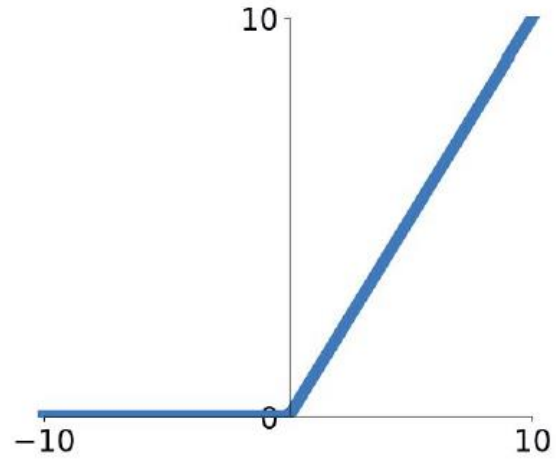
- Squashes numbers to range [0,1]
- Was very popular
- Problems
  - No gradient propagation at the saturated region
  - Its outputs are not zero-centered
  - e^{-x} is a bit expensive to compute

Activation Functions: Tanh



- Squashes numbers to range [0,1]
- Zero-centered
- No gradient propagation at the saturated region

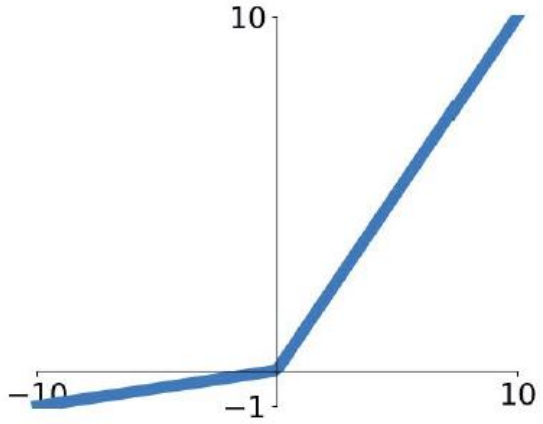
Activation Functions: ReLU



ReLU
(Rectified Linear Unit)

- $f(x) = \max(0, x)$
- Does not saturate in positive region
- Easy to compute
- Converges much faster than sigmoid/tanh in practice
- More biological plausible than sigmoid
- Problem
  - Not zero-centered output
  - No gradient when $x < 0$, ReLU will not activate

Activation Functions: Leaky ReLU

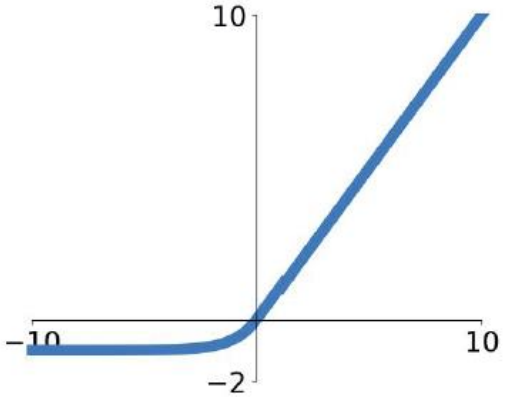


Leaky ReLU

$$f(x) = \max(0.01x, x)$$

- Does not saturate in positive region
- Easy to compute
- Converges much faster than sigmoid/tanh in practice
- More biological plausible than sigmoid
- Will not die

Activation Functions: Exponential Linear Units (ELU)



$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha\,(\exp(x) - 1) & \text{if } x \leq 0 \end{cases}$$

- All benefits of ReLU
- Mean outputs are closer to zero
- Negative region is more robust to noise compared to Leaky ReLU
- Can be expensive as it might require e^x computation

Weight Initialization

- How to set the first values of weights?
- With proper data normalization, it is reasonable to assume that half of the weights will be positive and half of them will be negative, or expected value of weights is zero but:
  - Avoid all zero initialization because backward gradient propagation is zero
  - Initialize to small random numbers so that the expected value of weight is close to 0
  - It is recommended to use the following initialization
  - $w = \frac{\text{randn}(n)}{\text{sqrt}(n)}$ where n is the number of its inputs.