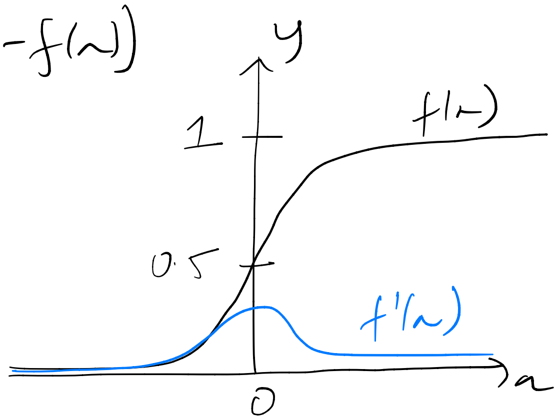
* Sigmoid Activation function

$$f(x) = \frac{1}{1+e^{-x}} \qquad f'(x) = f(x)(1-f(x))$$



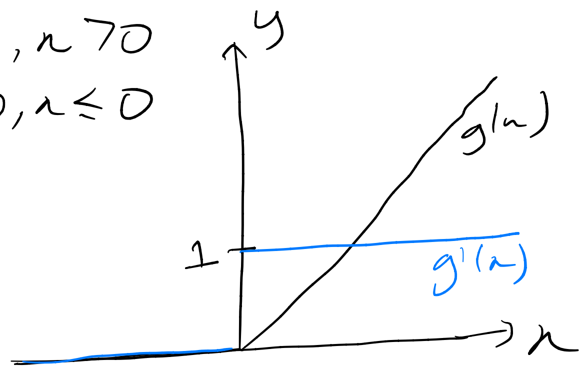
→ $f: \mathbb{R} \to \mathbb{R}^{(0,1)}$

→ bounds the final output between 0 to 1.

→ useful for classification problem.

* ReLU (Rectified Linear Units) activation funct.

$$g(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases} \qquad g'(x) = \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases}$$



→ $g: \mathbb{R} \to \mathbb{R}^{+}$

→ linear in positive dimension but zero in the negative dimension

→ Kink at $x=0$ is the source of non-linearity

ReLU vs Sigmoid

* ReLU over Sigmoid

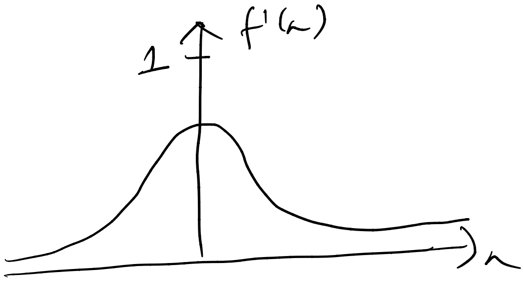
ReLU is preferred over Sigmoid for deep neural networks for the following reasons:

i) With ReLU, there is no vanishing gradient problem when compared to sigmoid function.

Sigmoid has a mechanism to reduce the gradient as the input (x) increases.

Gradient of Sigmoid:

$$f'(x) = f(x)\,(1 - f(x))$$



Then, when $x \to \infty$, $f(x) \to 1$

hence, $f'(x) \to 1\,(1-1)$
$= 0$

ii) ReLU is computationally more efficient to compute than sigmoid since ReLU just needs to pick $\max(0, x)$ and not perform exponential operations as in sigmoid.

iii) ReLU tend to show better convergence performance than sigmoid.

* Sigmoid over ReLU

i) ReLU tends to blow up activation as there is no mechanism to constraint the output of neuron. Sigmoid can constrant the output value.

ii) Dying ReLU problem: if too many activations get below zero, then most of the unit (neurons) in the network with ReLU will simply output zero. This prohibits learning. This problem is less likely to happen with sigmoid. But to tackle this problem and maintain the properties of ReLU, leaky-ReLU was proposed.

Leaky-ReLU

$$h(x) = \begin{cases} x, & x > 0 \\ \alpha x, & x \leq 0 \end{cases}$$ where α is a small constant.

$$h'(x) = \begin{cases} 1, & x > 0 \\ \alpha, & x \leq 0 \end{cases}$$