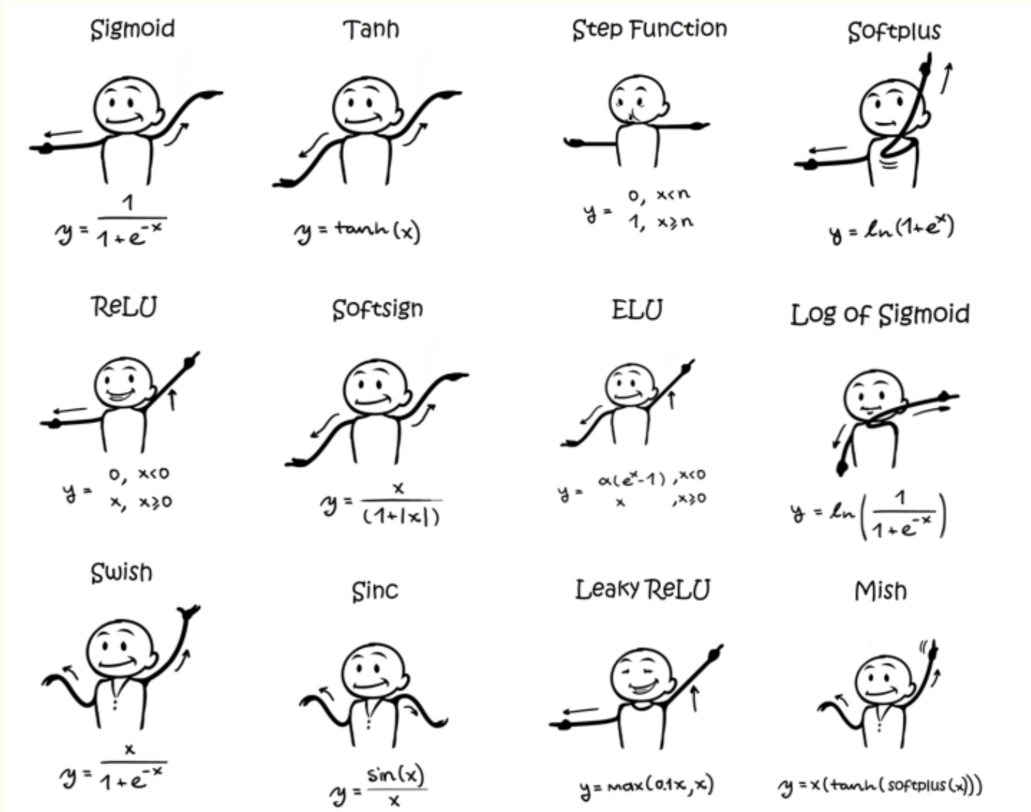
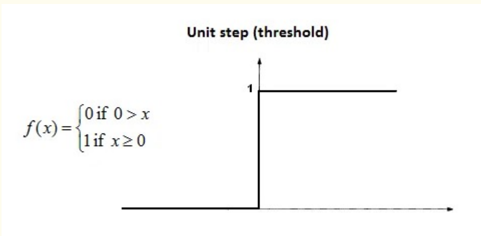


1. Step Function:



adv:

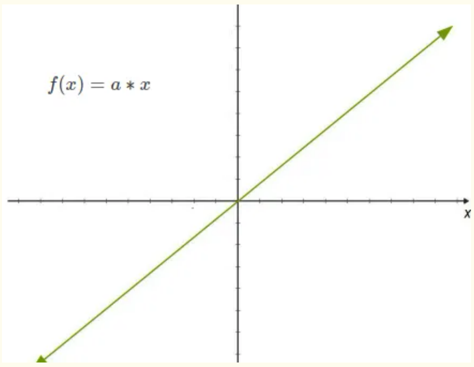
→ works well for binary classification

disadv:

→ gradient of function is = 0. during back-propagation weight updatation will not happen.

→ can not be used for multiclass classification

2. Linear Function:



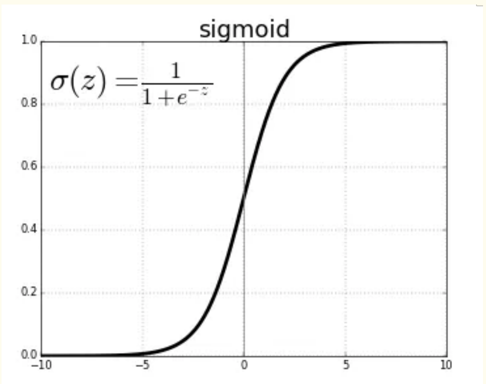
adv:

→ highly interpretable

disadv:

→ derivative of linear function = const. so during backpropagation weight updatation will be same. may lead to overshoot from optimal solution.

3. Sigmoid:



adv:

→ used as the output of binary probabilistic function.

→ smooth function & continuously differen-tiable.

→ one of the best Normalised functions.

disadv:

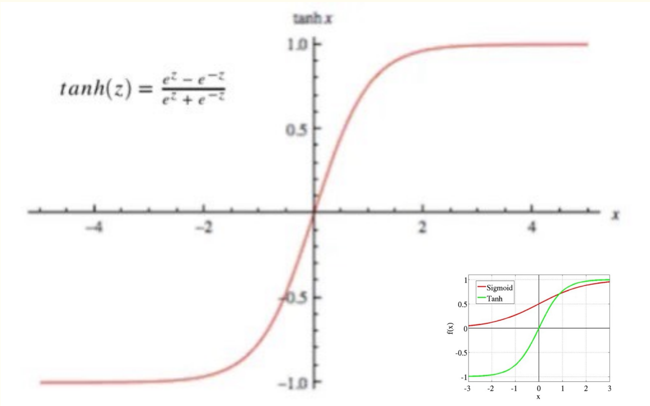
→ suffer from Vanishing gradient problem. saturate and kills gradient.

→ slow convergence

→ Not a zero-centric function, always gives a +ve values. gradient updates go too far in different directions.

→ computationally expensive function (exponential in nature)

4. **tanh:**



adv:

→ binary probabilistic function.

→ zero-centric function unlike sigmoid.

→ smooth gradient converging function

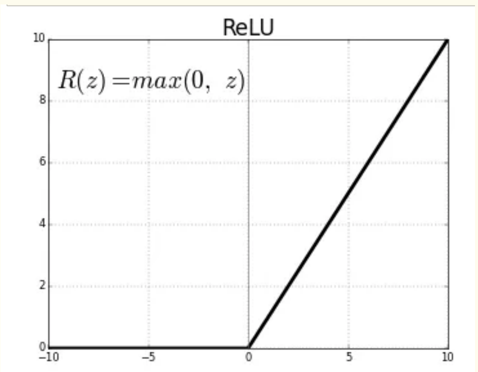
→ Brings non-linearity to the models.

disadv:

→ prone to vanishing gradient.

→ computationally expensive.

5. **ReLU:**



$$RELU(x) = \begin{cases} 0 \text{ if } x < 0 \\ x \text{ if } x >= 0 \end{cases}$$

adv:

→ computationally inexpensive.

→ can handle Vanishing Gradient problem.

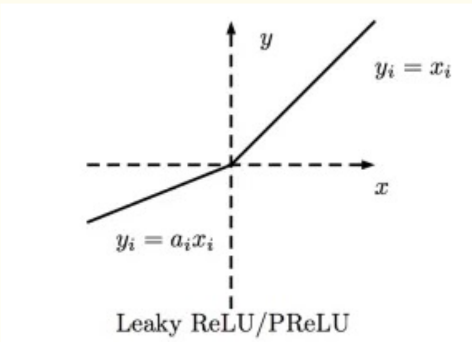
→ greatly accelerate convergance in SGD.

→ does not activate all the neurons at the same time. some are zero. brings sparsity to the model.

disadv:

→ not differentiable at zero.
→ ReLU is unbounded.
→ gradients for -ve input = 0. so weights can not be updated during back-propagation.
↳ leads to dead activation.
→ may lead to Explosive Gradient problem.

6. Leaky - ReLU:



$\alpha = 0.01$

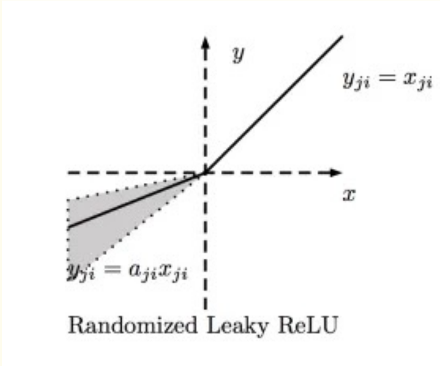
adv:

→ does not suffer from dead activation as gradient exist for -ve input.

disadv:

→ saturate for large negative values. may lead to inactive.
→ practically Leaky ReLU not always better than ReLU.
↳ only to be used when see lots of dead neurons.

7. Randomized Leaky ReLU:

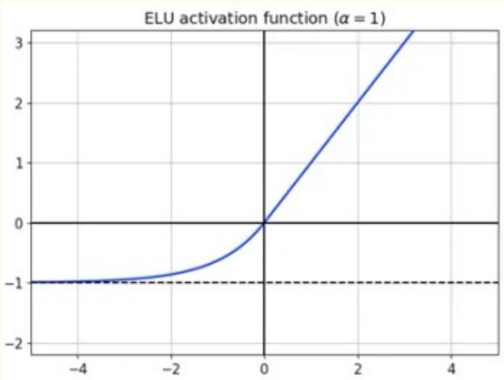


$$y_{ji} = \begin{cases} x_{ji} & \text{if } x_{ji} \geq 0 \\ a_{ji}x_{ji} & \text{if } x_{ji} < 0, \end{cases}$$

where

$$a_{ji} \sim U(l, u), l < u \text{ and } l, u \in [0, 1)$$

8. ELU and SELU:

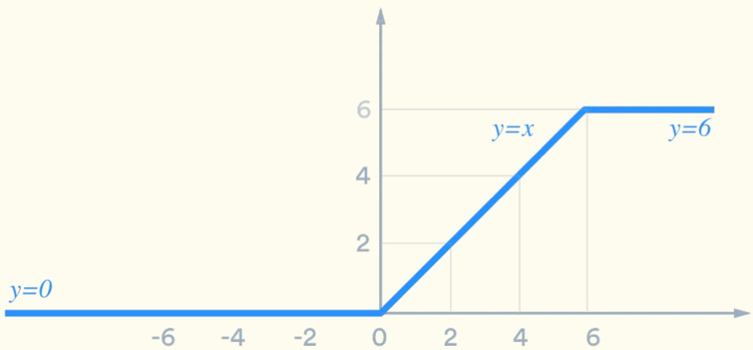


$$f(\alpha, x) = \begin{cases} \alpha(e^x - 1) & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases}$$

$$f(\alpha, x) = \lambda \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

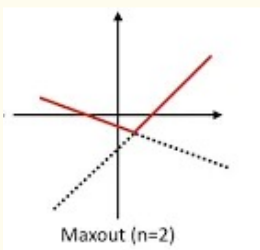
with $\lambda = 1.0507$ and $\alpha = 1.67326$

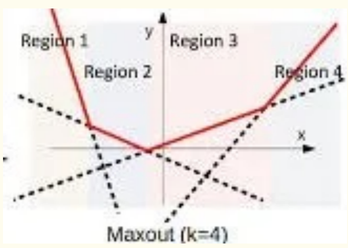
9. ReLU-6:



10. Maxout:

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





11. SWISH:

→ self gated function
→ used in LSTM.
→ can deal with vanishing gradient problem.
→ output is workaround b/w ReLU and sigmoid function → helps in normalising the output.

disadv:

→ computationally expensive.

$$f = \sum w_i x_i + b$$

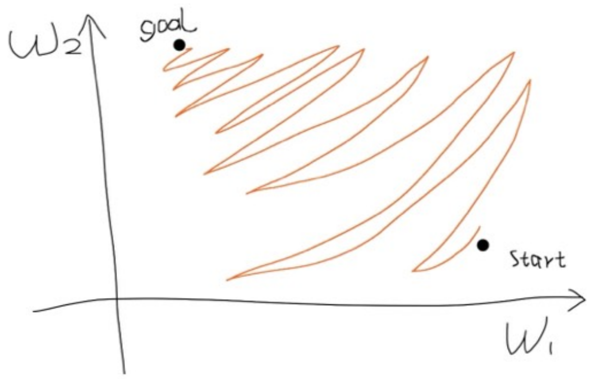
$$\frac{df}{dw_i} = x_i$$

$$\frac{dL}{dw_i} = \frac{dL}{df}\frac{df}{dw_i} = \frac{dL}{df}x_i$$

because $x_i > 0$, the gradient $\frac{dL}{dw_i}$ always has the same sign as $\frac{dL}{df}$ (all positive or all negative).

Update
Say there are two parameters w_1 and w_2. If the gradients of two dimensions are always of the same sign (i.e., either both are positive or both are negative), it means we can only move roughly in the direction of northeast or southwest in the parameter space.

If our goal happens to be in the northwest, we can only move in a zig-zagging fashion to get there, just like parallel parking in a narrow space. (forgive my drawing)



Therefore all-positive or all-negative activation functions (relu, sigmoid) can be difficult for gradient based optimization. To solve this problem we can normalize the data in advance to be zero-centered as in batch/layer normalization.

Also another solution I can think of is to add a bias term for each input so the layer becomes