Weight Initialization

1) Define your nn architecture, input-dim, HL → neurons
   o/p, activation, bias, matrices

2) weights, and biases in the nn → randomly initialized

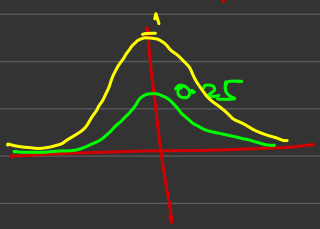
3) Training → FPE BP, the weights get trained

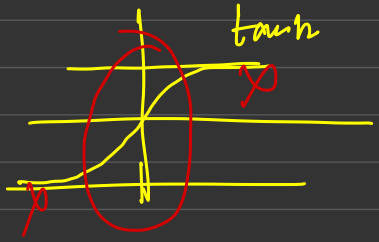
Zero Initialization :
↳ the entire network becomes symmetrical
→ ✗

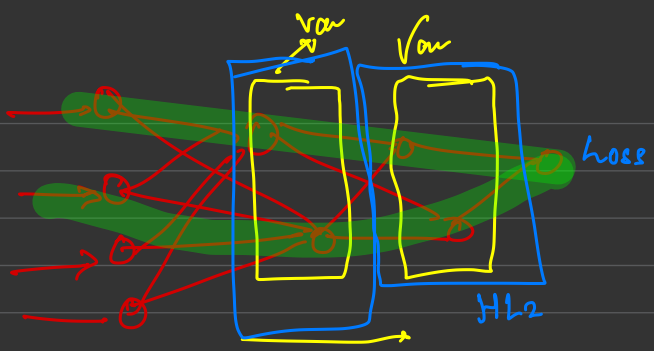
Random Initialization
↳ Generate weights which are very high
↳ activations high

Sigmoid

very low ← Vanishing gradient descent







forward flowing signals

$$Var\left(\frac{dh}{dHL2}\right) = Var\left(\frac{dh}{dHL1}\right)$$

to maintain that the neural network does not enter into a Vanishing or Exploding gradient problem.

Xavier initialization
He initialization

Xavier initialization (tanh)

$$W^{[l]} \sim N\left(\frac{N(0,1)}{n_{i+1}, n_i}\right) \times \sqrt{\frac{1}{n_i}}$$

n_i = n_{i+1}

$n_{i+1} \times n_i$

$$W^{[l]} \sim N(n_{i+1}, n_i) \times \sqrt{\frac{2}{n_{i+1}+n_i}}$$

Normalized Xavier initialization

For Relu activation

He initialization:

$N(0, \sigma^2)$

$W^{[l]} \sim N(n_{i+1}, n_i) \times \sqrt{\frac{2}{n_i}}$