Deep Learning

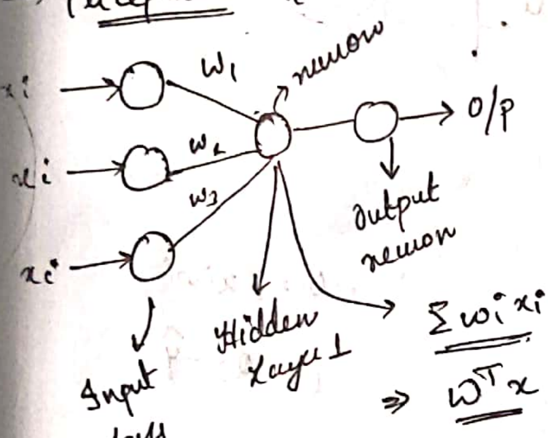
1. Forward and Backward propogation
2. Loss Function
3. Activation functions
4. Optimizers.

→ Deep learning tries to mimic human brain – Multi Layered neural network

→ Deep Learning is becoming popular because of ① huge amount of data,
② Hardware advancement (GPUs)
③

→ Perceptron:- { single Layer neural network }

w_1, w_2, w_3 } weights → weights say that how much a neuron should get activated or deactivated



$\left.\begin{array}{l}\Sigma w_i x_i \\ \Rightarrow w^T x\end{array}\right\}$ step 1

step 2:- Pass this to a activation function

→ A Bias is added (constant) as if all ~~intercepts~~ weights are 0, then also the training proceeds.

Sigmoid activation function $\Rightarrow \frac{1}{1+e^{-y}} \Rightarrow \frac{1}{1+e^{-(\Sigma x_i w_i + b)}}$

$\Rightarrow Act(y) \longrightarrow$ then the output

→ This entire process is called forward propogation

→ $(y - \hat{y})$ should be equal to almost 0. $(y-\hat{y})$ is also called as loss function. If difference is huge we do back propogation.

→ Optimizers will help to update the weights while back propogation.

→ Best known example of optimizer is Gradient descent.

---

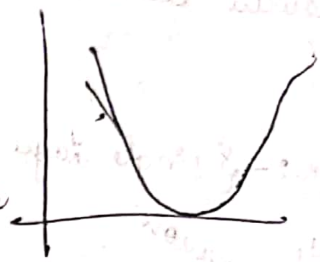
① Vanishing gradient problem
② Loss function

Backpropogation

① weight updation formula
② Chain Rule of Diffentation

$$W_{new} = W_{old} - \eta \frac{\partial L}{\partial W_{old}}$$

↳ Learning Rate

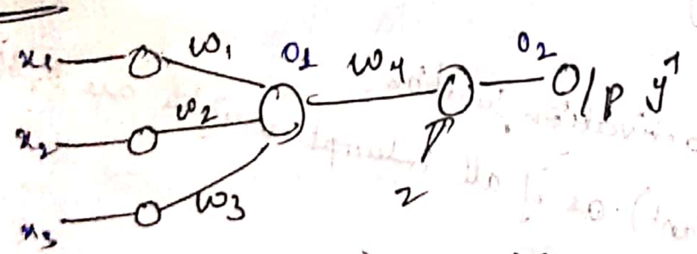


$$\frac{\partial L}{\partial W_{old}} = \text{slope} \qquad \eta = 0.01$$

Chain Rule of Diffentiation

$O_1 = \sum x_i w_i$

$L = (y - \hat{y})$



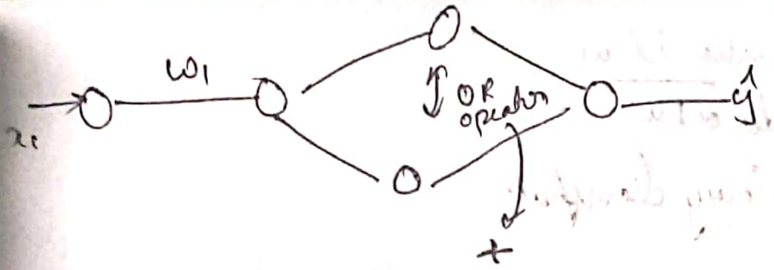
$Z = \sigma(O_1 w_4 + b)$

$$\frac{\partial L}{\partial W_{4old}} = \frac{\partial L}{\partial O_2} \times \frac{\partial O_2}{\partial W_{4old}}$$

$$W_{4new} = W_{4old} - \eta \frac{\partial L}{\partial W_{4old}}$$

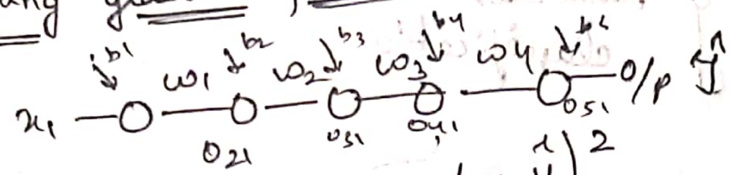
$$b_{2new} = b_{2old} - \eta \frac{\partial L}{\partial b_{2old}}$$

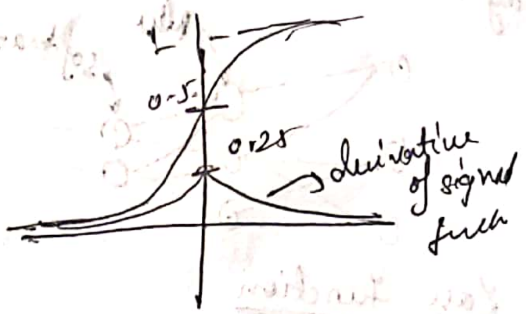
$$\frac{\partial L}{\partial W_{1old}} = \frac{\partial L}{\partial O_2} \times \frac{\partial O_2}{\partial O_1} \times \frac{\partial O_1}{\partial W_{1old}}$$



add both paths

Vanishing gradient problem





LOSS = $\frac{1}{2n}(y-\hat{y})^2$
(Mean squared error) (MSE)

$$\frac{\partial h}{\partial w_{1new}} = \frac{\partial h}{\partial O_{51}} \times \frac{\partial O_{51}}{\partial O_{41}} \times \frac{\partial O_{41}}{\partial O_{31}} \times \frac{\partial O_{31}}{\partial O_{21}} \times \frac{\partial O_{21}}{\partial w_{1new}}$$

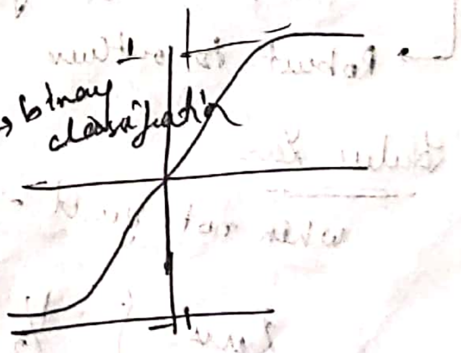
$$w_{1new} = w_{1old} - \eta \times \frac{\partial L}{\partial w_{1new}}$$

$$O_{51} = \sigma(O_{41} \times w_4 + b)$$
↳ Activation fn

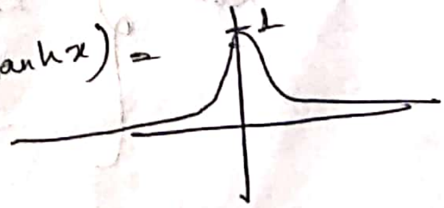
when $w_{new} \approx w_{old}$ ⇒ vanishing gradient problem

Activation functions

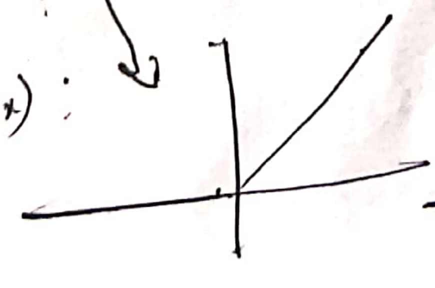
① Sigmoid → O/p layer → binary classification
② Tanh → $\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ → for hidden layer

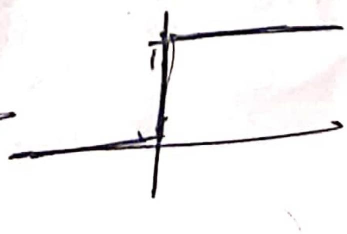


$d(\tanh x) =$



③ Relu → $\max(0, x)$
④ Leaky Relu → $\max(0.01x, x)$
⑤ Para relu



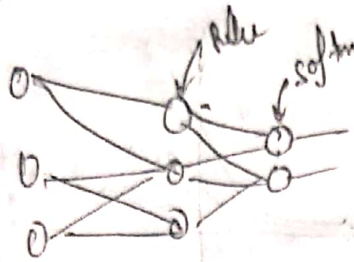


Technique which Activation f^n we should use

In hidden layer - Relu → PreRelu or Elu } Binary classification

O/p layer - Sigmoid

definite



} Multiclass classification

Loss Function.

~~Deep Learning~~	Loss f^n	Cost function
① Regression		
→ Mean Square Error	$\frac{1}{2}(y-\hat{y})^2$	$\frac{1}{2n}\sum_{i=1}^{n}(y-\hat{y})^2$
→ Mean Absolute Error	$\frac{1}{2}\lvert y-\hat{y}\rvert$	$\frac{1}{2n}\sum_{i=1}^{n}\lvert y-\hat{y}\rvert$
→ Huber Loss		

→ Advantages

① Differentiable

② only one global/local minima

③ It converges faster

Disadvantages

① Not Robust to outliers

→ Robust to outliers

② Huber Loss

when not present - outliers

$$\text{Loss} = \begin{cases} \frac{1}{2}(y-\hat{y})^2 & \text{if } |y-\hat{y}| \leq \delta \\ \delta|y-\hat{y}| - \frac{1}{2}\delta^2, & \text{otherwise.} \end{cases}$$

Classification

↳ Cross Entropy → Binary cross Entropy

→ Categorical cross entropy

Same Loss function as that of Logistic Regression

→ Do one hot encoding

$$L(x_i, y_i) = -\sum_{j=1}^{c} y_{ij} \times \ln(\hat{y}_{ij})$$

$$y_{ij} = \begin{cases} 1 & \text{if the element is in class } j \\ 0 & \text{otherwise} \end{cases}$$

Softmax activation

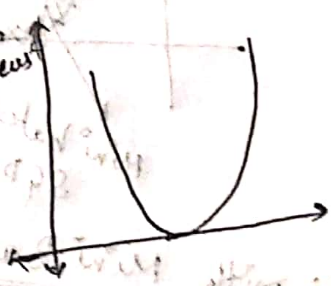
$$\sigma(z) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$$

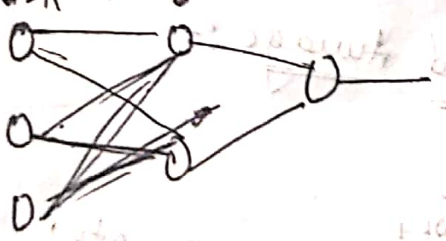
Optimizers

→ ① Gradient Descent

weight updation formula:

$$\boxed{W_{new} = W_{old} - \eta \frac{\partial L}{\partial W_{old}}}$$





$$\text{Loss function} = \frac{1}{2n} \sum_{i=1}^{n} (y - \hat{y})^2$$

Disadvantage

① Resource extensive task (Huge RAM)

Epoch

1 FP and BP → 1 epoch

② **Stochastic Gradient Descent**

→ We will pass datapoints in iterations

→ Same like Gradient Descent

→ Ram required is less but the converge will be very slow.

→ High Time complexity

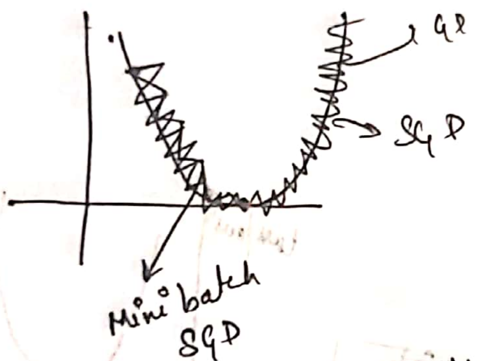
③ **Mini Batch Stochastic Gradient Descent**

→ In batches, one batch might have 1000 reads

→ Resource intensive

→ Converge rate will be better

→ Time complexity will improve



④ ~~SGD with~~ **Mini Batch SGD with Momentum**

→ Momentum will help to reduce the noise

Exponential Moving/Weighted Average :-

$$w_t = w_{t-1} - \eta \frac{\partial h}{\partial w_{t-1}}$$

$\beta = 0$ to 1
↳ Hyperparameter

$$V_{t_1} = a_1$$

$$V_{t_2} = \beta \times V_{t_1} + (1-\beta) \times a_2$$

$$V_{t_3} = \beta \times V_{t_2} + (1-\beta) \times a_3$$

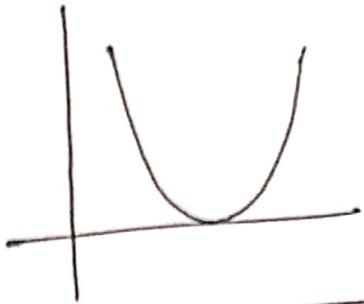
$$w_t = w_{t-1} - \eta \, V_{dw}$$

$$\boxed{V_{dw} = \beta \times V_{dw_{t-1}} + (1-\beta) \times \frac{\partial L}{\partial w_{t-1}}}$$

(5) Adagrad - Adaptive Gradient Descent

→ Learning Rate is not constant

$$\boxed{w_t = w_{t-1} - \eta' \frac{\partial L}{\partial w_{t-1}}}$$



$$\eta' = \frac{\eta}{\sqrt{\alpha_t + \epsilon}}$$

$\eta \rightarrow$ decrease it

→ might become very less

$\boxed{\epsilon = \text{small number to avoid divide by 0 condition}}$

$$\alpha_t = \sum_{i=1}^{t} \left(\frac{\partial L}{\partial w_t}\right)^2$$

→ we should see α_t is never so big

(6) Adadelta and RMS prop

$$\eta' = \frac{\eta}{\sqrt{s_{dw} + \epsilon}}$$

← Exponential weighted average

$$s_{dw_t} = \beta \times s_{dw_{t-1}} + (1-\beta)\left(\frac{\partial h}{\partial t_4}\right)^2$$

$$(s_{dw})_t = 0$$

(7) Adam Optimizer

Momentum + RMSprop

$$V_{dw} = \beta \times V_{dw_{t-1}} + (1-\beta)\frac{\partial L}{\partial w_{t-1}}$$

$$w_t = w_{t-1} - \eta' \, V_{dw}$$