Pbr:

Assume we are given a dataset of N observations

Input: $x^1, x^2, x^3 \dots x^N$ $\{x^i\}$ $x^i \in \mathbb{R}^n \rightarrow$ Vector

Output: $y^1, y^2, y^3 \dots y^N$ $\{y^i\}$ $y^i \in \mathbb{R} \rightarrow$ Scalar

Straight x $\quad x^i = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ Each x^i is a vector

Cursive x

Assume that the data was generated by some function

$$f(x) : \mathbb{R}^n \rightarrow \mathbb{R}$$

[Not known. Maybe no closed form. Eg Image, Sound clip]

We try to model f by an approximation $\hat{f}$

$$\hat{f}(x, w) = \underbrace{w_0}_{\text{Parameters}} + \sum_{j=1}^{m-1} w_j \underbrace{\phi_j(x)}_{\text{Basis functions}}$$

Linear in Parameters. Basis functions can be non-linear.

$$W = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_m \end{bmatrix} \in \mathbb{R}^m \rightarrow \text{Parameter vector}$$

$w_0 \rightarrow$ Bias parameter

$\phi_0(x) = 1$

$$\phi v(x) = \begin{bmatrix} \phi_0(x) \\ \phi_1(x) \\ \vdots \\ \phi_m(x) \end{bmatrix} \rightarrow \text{Feature vector} \in \mathbb{R}^m$$

$$\hat{y} = \hat{f}(x, w) = \phi v^T_{(x)} \cdot w$$

Aim: Find w s.t. $\hat{f}(x, w)$ is close to f for all data points.

Examples:

1. Linear Basis: $x \in \mathbb{R}^n$ $\quad m = n+1$

$$\phi_j(x) = x_j \quad j \in [0 \dots n] \quad \phi_0(x) = 1$$

- Let $x \in \mathbb{R}^2 = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

$$\phi_0(x) = 1, \quad \phi_1(x) = x_1 \quad \phi_2(x) = x_2$$

$$\hat{f}(x, w) = w_0 + w_1 x_1 + w_2 x_2$$

2. Polynomial Basis: $x \in \mathbb{R}$

$$\phi_j(x) = x^j$$

Let $m = 2$

$$\phi_0(x) = 1, \; \phi_1(x) = x, \; \phi_2(x) = x^2$$

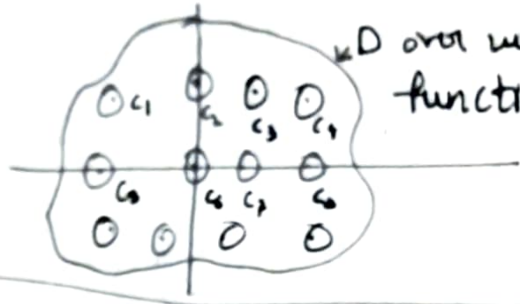
$$\hat{f}(x, w) = w_0 + w_1 x + w_2 x^2$$

3. Radial Basis functions: $x \in \mathbb{R}^n$, $\phi_j(x) = \exp\left(\frac{\|x - c_j\|_2}{\sigma_j^2}\right)^2$

Centers: $c_j \in \mathbb{R}^n$

Let $x \in \mathbb{R}^2 = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

$m = 2$



D over which we want to approximate functions

Aim: Find w s.t $\hat{f}$ and f are close for each data point.

Sum of squares Error:

(Sample by sample)

$$E(w) = \frac{1}{N} \sum_{i=1}^{N} \{ y^i - \hat{y}^i \}^2$$

Actual data

Data predicted by $\tilde{f}$

$$= \frac{1}{N} \sum_{i=1}^{N} \{ y^i - \phi^T(x^i) \cdot w \}$$

Select w s.t. E is minimized

Gradient descent

Calculate gradient of $E(w)$.

Vectors: Regression Matrix

$$\Phi = \begin{bmatrix} \phi^v(x^1)^T \\ \phi^v(x^2)^T \\ \vdots \\ \phi^v(x^N)^T \end{bmatrix} \in \mathbb{R}^{N \times m} \quad Y = \begin{bmatrix} y^1 \\ y^2 \\ \vdots \\ y^N \end{bmatrix} \in \mathbb{R}^N$$

$$\hat{y} = \begin{bmatrix} \hat{y}^1 \\ \vdots \\ \hat{y}^N \end{bmatrix} = \begin{bmatrix} \phi_v(x^1)^T \cdot w \\ \phi_v(x^2)^T \cdot w \\ \vdots \\ \phi_v(x^N)^T \cdot w \end{bmatrix} = \begin{bmatrix} \phi_v(x^1)^T \\ \phi_v(x^2)^T \\ \vdots \\ \phi_v(x^N)^T \end{bmatrix} w = \Phi w \in \mathbb{R}^N$$

$$E(w) = \frac{1}{N}\left(y^i - \hat{y}^i\right)^2 = \frac{1}{N}\|y - \hat{y}\|_2^2$$

$$= \frac{1}{N}(y - \hat{y})^T (y - \hat{y}) \qquad \text{where } \hat{y} = \Phi w$$

$$= \frac{1}{N}\left(y^T y - y^T \Phi w - w^T \Phi^T y + w^T \Phi^T \Phi w\right)$$

$$\mathbb{R}^m \ni \nabla E(w) = \frac{1}{N}\left(-\Phi^T y - \Phi^T y + 2\Phi^T \Phi w\right)$$

since $w \in \mathbb{R}^m$

$$= \frac{2}{N}\left(-\Phi^T y + \Phi^T \Phi w\right)$$

Gradient is a function of w

1] Gradient descent:

Constant step size : η
(Learning rate)

Algo: Given $\{x^i\}$ $\{y^i\}$ → Data set

Compute Φ, and Y

$W = \text{rand}(m)$ → Initialize $W \in \mathbb{R}^m$ randomly

for $i = 1$: max_steps

$$\nabla E = \frac{2}{N}\left(-\Phi^T Y + \Phi^T \Phi W\right)$$

$$W = W - \eta \nabla E$$

end

Problems: 1. Requires all the data to calculate gradient.
2. What if data is so large that it does not fit in memory

2] Stochastic grdient descent:

Idea: 1. Do not ~~look~~ use whole data

2. Use one sample at a time

3. Go through the data sequentially.

$$L(w) = \frac{1}{N}\left(y^i - \phi_v(x^i)^T \cdot w\right)^2$$

$$\nabla L = \cancel{\frac{2}{N}\left(-\Phi^T[i] \cdot Y[i] - \Phi_v[i]^T \Phi[i] w\right)}$$

$$= \frac{2}{N}\left(y^i - \phi_v(x_i)^T w\right) \cdot \phi_v(x^i)$$

Algo: Given

$\{x^i\}$ and $\{y^i\}$ → Data set

~~Compute~~ $W = \text{rand}(m)$

for $i = 1$: max_steps → [number of epoch]

for $j = 1 : N$

$$\nabla L = \frac{2}{N}\left(y^i - \phi_v(x_i)^T w\right) \cdot \phi_v(x^i)$$

end

end

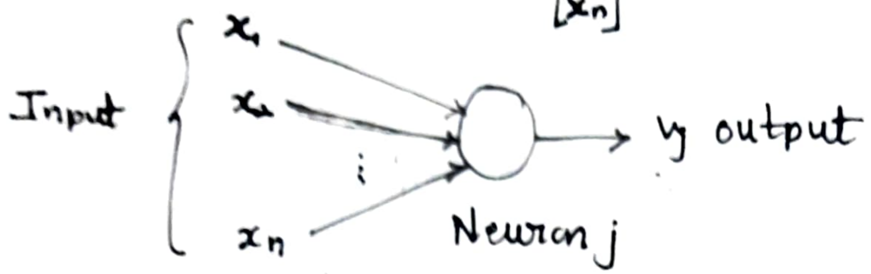
One gradient step on the average. ↓ epoch

3] Batch gradient descent: In between. [$\bar{N}$ sized batches of data

$$L(w) = \frac{1}{N} \sum_{j=1}^{\bar{N}} \left(y^i - \phi_v(x^i)^T \cdot w\right)$$

Neural Networks

Neuron: Input: $x \in \mathbb{R}^n = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$ Output $v_j \in \mathbb{R}$



$$h_j = \sum_{i=1}^{n} w_{ji} x_i + b_j \rightarrow \text{bias term}$$

$$v_j = f(h_j) \qquad f: \mathbb{R} \rightarrow \mathbb{R} \quad \text{Activation function}$$

Let $W_j = \begin{bmatrix} w_{j1} \\ w_{j2} \\ \vdots \\ w_{jn} \end{bmatrix} \in \mathbb{R}^n$

$$\therefore \; h_j = w_j^T x + b_i$$

$$v_j = f(h_j) = f(w_j^T x + b_i)$$

Examples of activation functions

1. Sigmoid (Logistic): $\sigma(x) = \dfrac{1}{1+e^{-x}}$

$$\sigma'(x) = \sigma(x)(1 - \sigma(x))$$

2. Hyperbolic tangent: (tanh): $\tanh(x) = \dfrac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$

$$\tanh'(x) = 1 - \tanh^2(x)$$

3. ReLU: $f(x) = x$ if $x \geq 0$ $\qquad f'(x) = 1$ if $x > 0$

$= 0$ if $x < 0$ $\qquad = 0$ if $x < 0$

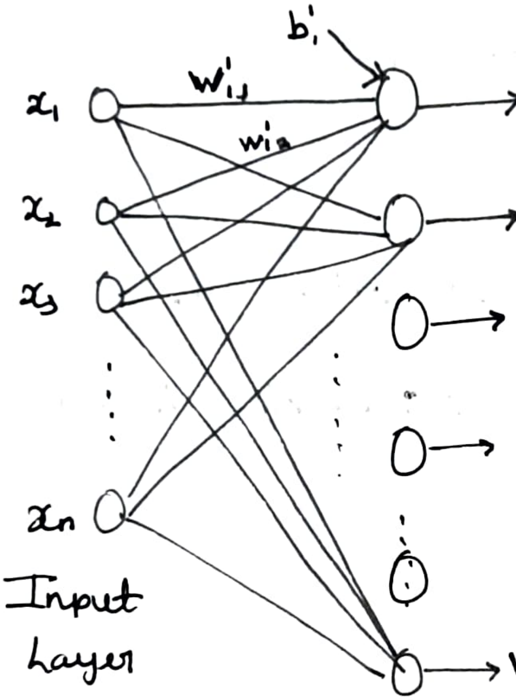
$f'(x) = 0$ at $x = 0$

Convention.

Neural Network: 1. Network of neurons.

2. Stack neurons on top of one another to form neural network

Input: $x \in \mathbb{R}^n$



$v_1' = f(W_1'^T x + b_1)$ (h_1)

$v_2' = f(W_2'^T x + b_2)$

$v_3' = f(W_3'^T x + b_3)$

v_4'

$v_{n_1}' = f(W_{n_1}'^T x + b_4)$

Layer 1

n_1 Neurons

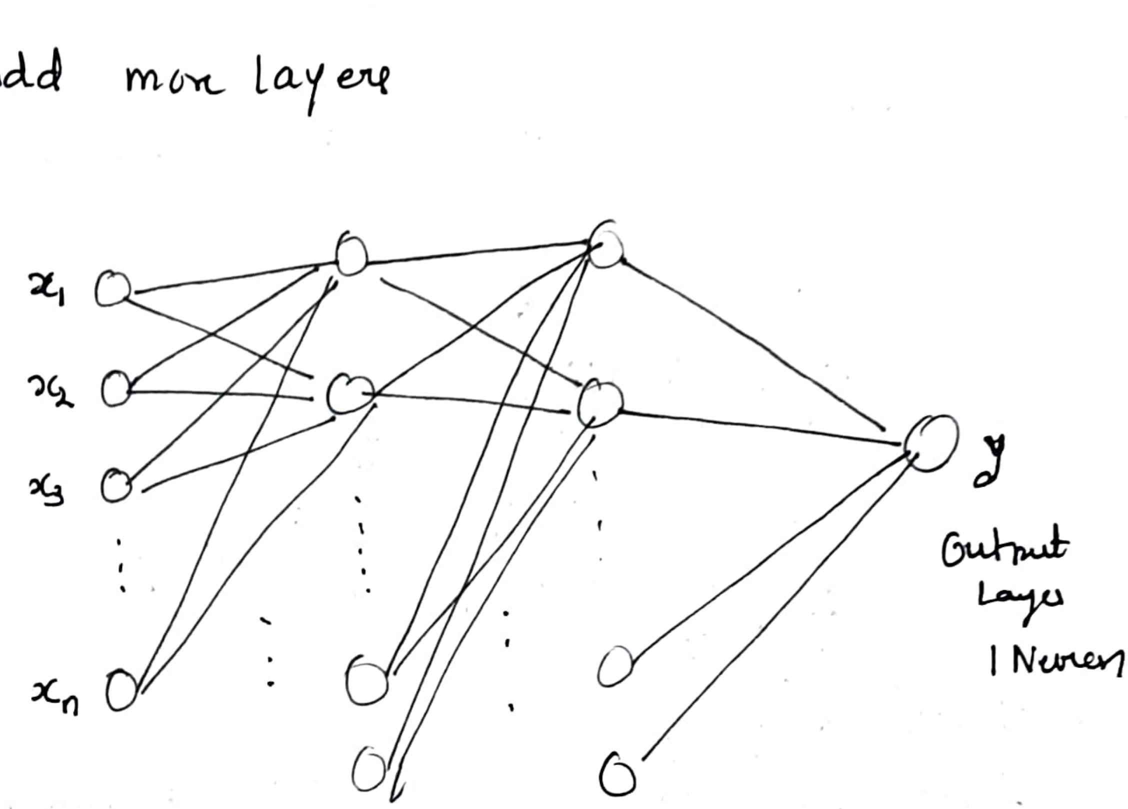
$$h_1 = \begin{bmatrix} h_1' \\ h_2' \\ \vdots \\ h_n' \end{bmatrix} = \begin{bmatrix} W_1'^T x + b_1 \\ W_2'^T x + b_2 \\ \vdots \\ W_n^T x + b_{n_1} \end{bmatrix} = \begin{bmatrix} W_1'^T \\ \vdots \\ W_{n_1}'^T \end{bmatrix} x + \begin{bmatrix} b_1' \\ b_2' \\ \vdots \\ b_{n_1}' \end{bmatrix}$$

$$h_1 = W_1 x + b_1$$

$$V_1 = \begin{bmatrix} f(h_1') \\ f(h_2') \\ \vdots \\ f(h_{n_1}') \end{bmatrix} = f\left(\begin{bmatrix} h_1' \\ \vdots \\ h_{n_1}' \end{bmatrix} \right) = f(h_1) \rightarrow \text{elementwise application.}$$

$$V_1 = f(W_1 x + b_1)$$

Add more layers



n_1 neurons	n_2 neurons	n_3 neurons
Layer 1	Layer 2	~~Out~~
$V_1 = f(W_1 x + b_1)$	$V_2 = f(W_2 V_1 + b_2)$	$V_3 = y = f(W_3 V_2 + b_3)$

$f = \text{Identity}$

$y = W_3 V_2 + b_3$

NN is a function:

$y = W_3 f(W_2 v_1 + b_1)$

$y = W_3 f(W_2 f(W_1 x + b_1))$

Parameters:

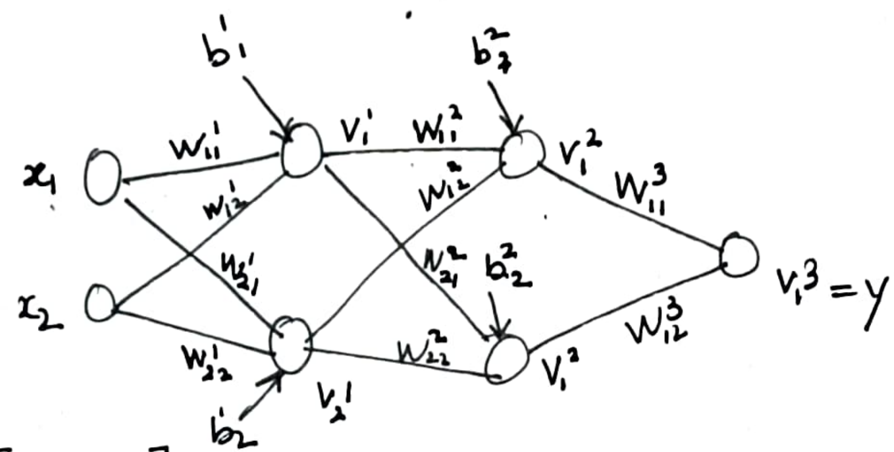
$P = [W_1, b_1, W_2, b_2, W_3, b_3]$

$y = F(x, P)$ → Compare to Linearly parametrized networks.

→ Parameters occure nonlinearly in output

Makes computing gradients a challenge.

eg] A simple NN with $x \in \mathbb{R}^2$, $n_1 = 2$, $n_2 = 2$, $n_3 = 1$



Step1] $h_1 = \begin{bmatrix} h_1^1 \\ h_2^1 \end{bmatrix} = \begin{bmatrix} W_{11}^1 & W_{12}^1 \\ W_{21}^1 & W_{22}^1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$ $\quad V_1 = \begin{bmatrix} v_1^1 \\ v_2^1 \end{bmatrix}$ } 6params $= n \times n_1 + n_1$

Step2] $h_2 = \begin{bmatrix} h_1^1 \\ h_2^1 \end{bmatrix} = \begin{bmatrix} W_{11}^2 & W_{12}^2 \\ W_{21}^2 & W_{22}^2 \end{bmatrix} \begin{bmatrix} \cancel{x_1}\,v_1^1 \\ \cancel{x_2}\,v_2^1 \end{bmatrix} + \begin{bmatrix} b_1^2 \\ b_2^2 \end{bmatrix}$ } 6 param $n_1 \times n_2 + n_2$

Step3] $y = v_1^3 = \begin{bmatrix} W_{11}^3 & W_{12}^3 \end{bmatrix} \begin{bmatrix} v_1^2 \\ v_2^2 \end{bmatrix} + \begin{bmatrix} b_1^3 \end{bmatrix}$ } 6 param $n_2 \times n_3 + 1$

$y = NN(x)$

$$E(P) = \frac{1}{N} \sum_{i=1}^{N} \left\{ \cancel{NN(x^i)}\, y^i - NN(x^i) \right\}^2$$

$$L^i = \frac{1}{N} L$$

$W_1 = W_1 + n \nabla E_i(P)_{W_1} \longrightarrow$ How to compute these gradient.

$W_2 = W_2 + n \nabla E_i(P)_{W_2}$

$\vdots$

Backpropogation: Computing derivatives Algorithmically

$W = L(z)$ be a function of z

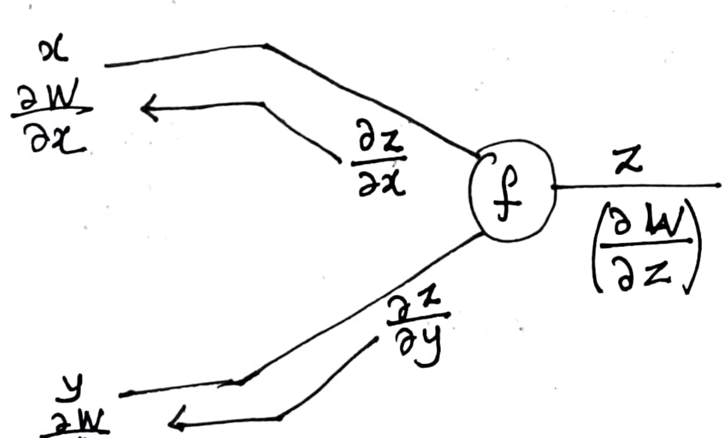
$z = f(x,y)$ be a function of x and y.

$$L(z) = L(f(x,y))$$

L is indirectly a function of x and y

$$\left.\begin{aligned} \therefore \frac{\partial W}{\partial x} &= \frac{\partial W}{\partial z} \cdot \frac{\partial z}{\partial x} \\ \frac{\partial W}{\partial y} &= \frac{\partial L}{\partial z} \cdot \frac{\partial z}{\partial y} \end{aligned}\right\} \text{Chain Rule.}$$

Computational graph of f:



Final
L → Output ~~function~~

1. Functions form nodes of computational graph.
2. x and y are input variables of node f
3. z is the output variable of node f.
3.1.
4. At each variable we also store derivative of output function L w.r.t that variable.

5. At each node compute local derivatives ~~w.r.t~~ (i.e. Derivative of output variable w.r.t input variable) and place it on the corresponding input branch

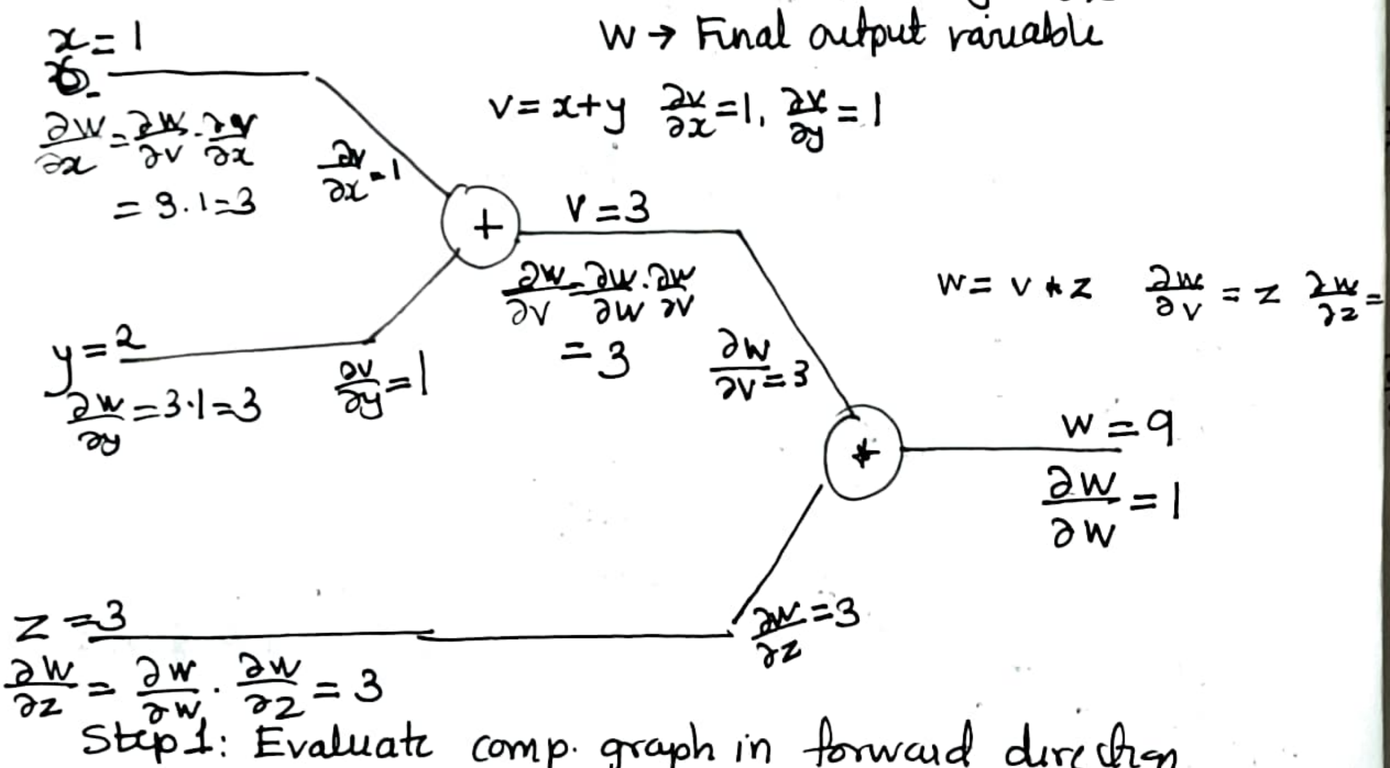
6. Folow the path from output variable to the input variable to get the corresponding input.

$$\frac{\partial w}{\partial x} = \frac{\partial w}{\partial z} \cdot \frac{\partial z}{\partial x}$$

$$\frac{\partial w}{\partial y} = \frac{\partial w}{\partial z} \cdot \frac{\partial z}{\partial y}$$

Ex. $w = (x+y) \cdot z$ We want $\frac{\partial w}{\partial x}, \frac{\partial w}{\partial y} \cdot \frac{\partial w}{\partial z}$

$w \rightarrow$ Final output variable



Step 1: Evaluate comp. graph in forward direction

Step 2: ~~Compute local derivative at all nodes~~ Start backprop

a) Start from end of C.G.

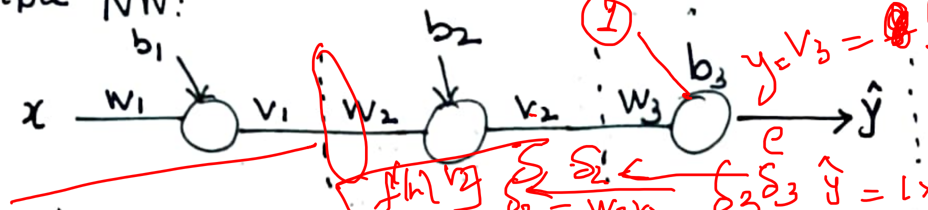
b) ~~Move backwards~~ From end compute derivative w.r.t each variable and local derivatives at each node

$v_1 = f(w_1 x + b_1)$

$v_2 = f(w_2 v_1 + b_2)$

$y = v_3 = f(w_3 v_2 + b_3) \to$ Linear

2] Simple NN:



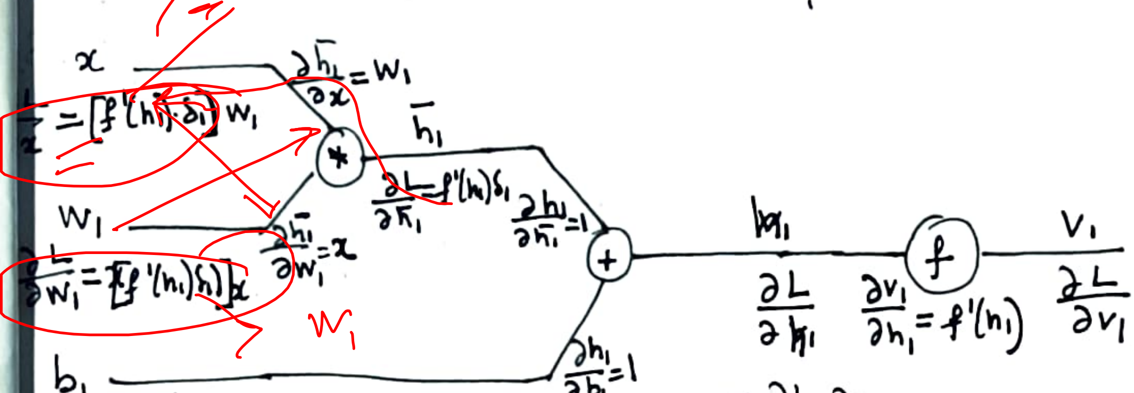
$f'(h)\, v_2 \quad \delta_2 \quad \delta_2' \quad \delta_3 \delta_3$

$\delta_2 = w_3 \delta_3$

$\hat{y} = 1 \times b + v_2 \cdot w_3$

$L = \frac{1}{2}(y - \hat{y})^2 = L(y)$

$L \to$ Final output variable



$\frac{\partial \bar{h}_1}{\partial x} = w_1$

$\frac{\partial L}{\partial x} = [f'(h_1)\delta_1]\, w_1$

$\frac{\partial \bar{h}_1}{\partial w_1} = x$

$\frac{\partial L}{\partial w_1} = [f'(h_1)\delta_1]\, x$

$\frac{\partial L}{\partial \bar{h}_1} = f'(h_1)\delta_1$

$\frac{\partial h_1}{\partial \bar{h}_1} = 1$

$\frac{\partial h_1}{\partial b_1} = 1$

$\frac{\partial L}{\partial b_1} = f'(h_1)\delta_1$

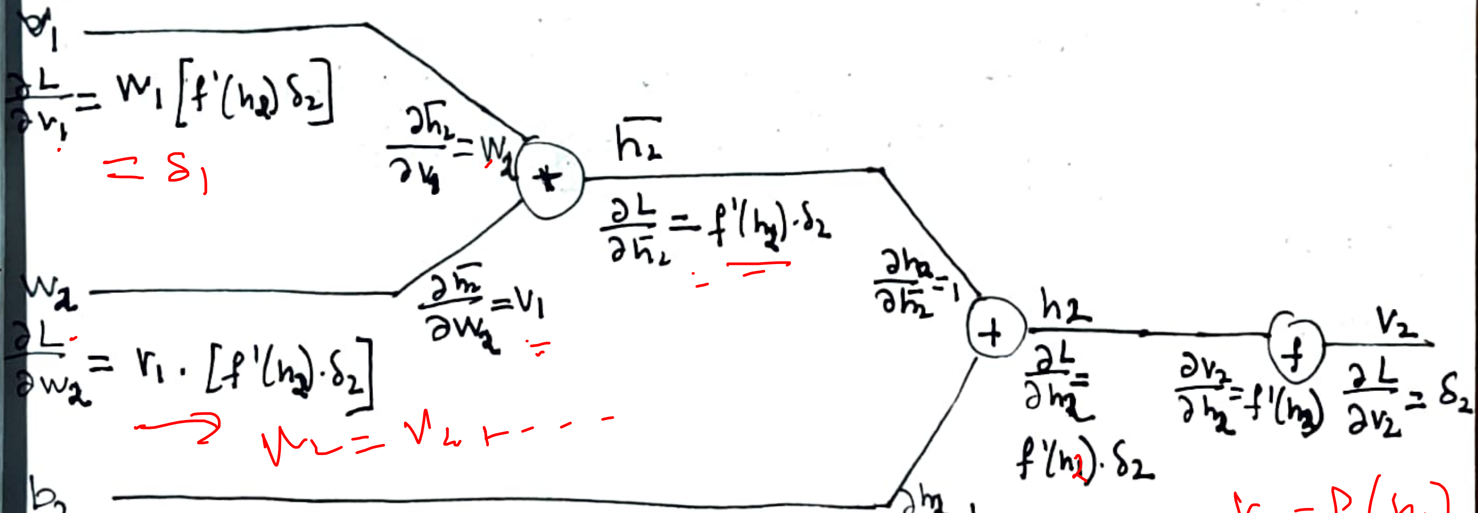
$\frac{\partial L}{\partial h_1}$ $\quad \frac{\partial v_1}{\partial h_1} = f'(h_1)$ $\quad \frac{\partial L}{\partial v_1} = \delta_1$

$= \frac{\partial L}{\partial v_1} \cdot \frac{\partial v_1}{\partial h_1}$

$= f'(h_1) \cdot \delta_1$

w_1

b_1



$\frac{\partial L}{\partial v_1} = w_1 [f'(h_2)\delta_2]$

$= \delta_1$

$\frac{\partial \bar{h}_2}{\partial v_1} = w_2$

$\frac{\partial L}{\partial \bar{h}_2} = f'(h_2) \cdot \delta_2$

$\frac{\partial h_2}{\partial \bar{h}_2} = 1$

$\frac{\partial \bar{h}_2}{\partial w_2} = v_1$

$\frac{\partial L}{\partial w_2} = v_1 \cdot [f'(h_2) \cdot \delta_2]$

$w_2 = w_2 - \dots$

$\frac{\partial L}{\partial h_2} = f'(h_2) \cdot \delta_2$

$\frac{\partial v_2}{\partial h_2} = f'(h_2)$ $\quad \frac{\partial L}{\partial v_2} = \delta_2$

$\frac{\partial h_2}{\partial b_2} = 1$

$\frac{\partial L}{\partial b_2} = f'(h_2) \cdot \delta_2$

$v_2 = f(h_2)$

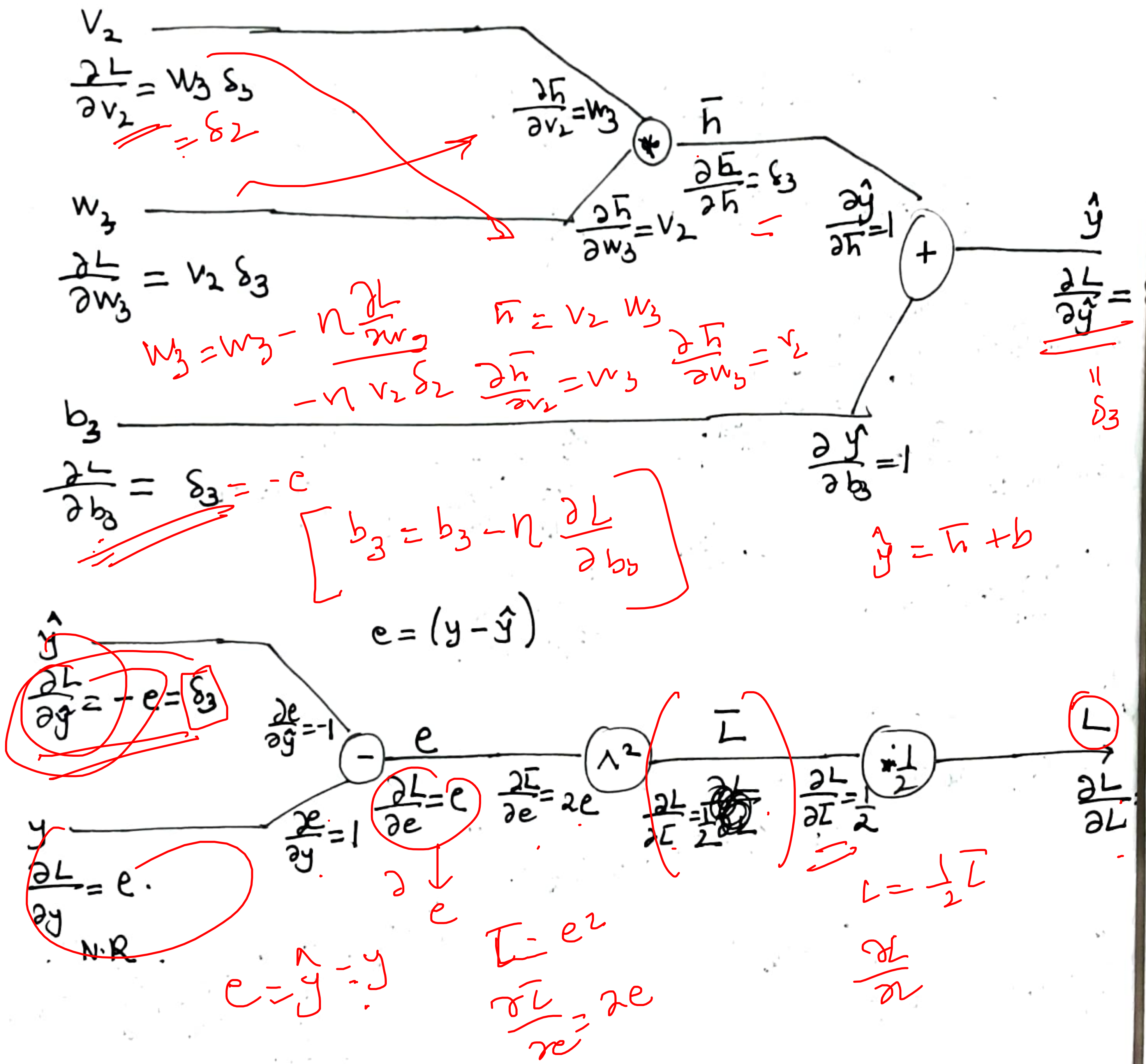
$h_2 = \bar{h}_2 + b_2$ $\quad \frac{\partial v_2}{\partial h_2} = f'(h_2)$

~~$b_2 = f'(h_2)$~~

$b_2 = b_2 - \eta\, f'(h_2) \cdot \delta_2$

$\hat{y} = w_3 v_2 + b_3$

$\bar{h}$



Algo: Compute Gradient

1. Forward pass

(Matrix Eq)

$$V_1 = f(W_1 x + b_1) \qquad v_1 = f(W_1 x + b_1)$$

$$V_2 = f(W_2 V_1 + b_2) \qquad V_2 = f(W_2 x + b_2)$$

$$V_3 = W_3 V_2 + b_3 = \hat{y} \qquad V_3 = W_3 V_2 + b_3 = \hat{y}$$

2. $\delta_3 = -(y^i - \hat{y}^i) = -e \qquad \delta_3 = -(y_i - \hat{y}^i) = -e$

$n_3 = 1$ $1 \times n_2$

$$\frac{\partial L}{\partial W_3} = [\delta_3] V_2 \qquad \mathbb{R}^{n_3 \times n_2}: \frac{\partial L}{\partial W_3} = \delta_3 \cdot V_2^T \quad 1 \times n$$

no of

$1 \times n$ $W_3 = W_3 - \eta$

$$\frac{\partial L}{\partial b} = \delta_3 \cdot 1 \qquad \mathbb{R}^{n_3}: \frac{\partial L}{\partial b_3} = \delta_3 \qquad W_3 = W_3 - \eta \frac{\partial L}{\partial W_3}$$

$\mathbb{R}^1$

$$\frac{\partial L}{\partial V_2} = \delta_2 = W_3 \delta_3 \qquad \mathbb{R}^{n_2}: \frac{\partial L}{\partial V_2} = W_3^T \cdot \delta_3 = \delta_2$$

$$\frac{\partial L}{\partial W_2} = [f_2'(h_2) \cdot \delta_2] \cdot V_1 \qquad \mathbb{R}^{n_2 \times n_1}: \partial L / \partial W_2 = [f_2'(h_2) \cdot \delta_2] \cdot V_1^T$$

$h_2 = W_2 V_1 + \cancel{b_2}$ output input

$$\frac{\partial L}{\partial b_2} = [f_2'(h_2) \cdot \delta_2] \qquad \mathbb{R}^{n_2}: \partial L / \partial b_2 = [f_2'(h_2) \cdot \delta_2]$$

Bays 1960

$$\frac{\partial L}{\partial v_1} = \delta_1 = W_2 [\delta_2 \cdot f_2'(h_2)] \qquad \mathbb{R}^{n_1}: \frac{\partial L}{\partial v_1} = W_2^T [f_2'(h_2) \cdot \delta_2] = \delta_1$$

$$\frac{\partial L}{\partial W_1} = [f_1'(h_1) \cdot \delta_1] \cdot x \qquad \mathbb{R}^{n_1 \times n}: \frac{\partial L}{\partial W_1} = [\delta_1 f_1'(h_1)]\, x^T$$

$$\frac{\partial L}{\partial b_1} = [\delta_1 f_1'(h_1)] \qquad \mathbb{R}^{n_1}: \frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial b_1} [\delta_1 f_1'(h_1)]$$