*) Backpropogation

•) Like the dependencies for loss on earlier parameters w and b were found and used to update them, in the same way the dependency of error/loss on each parameter of the net is evaluated and used to tune the parameters to increase accuracy. for example

$$\Delta w_{111} = \frac{\partial L(\theta)}{\partial w_{111}}$$

unit in the layer 1

unit in the layer 2

corresponding layer

$$\Delta b_{12} = \frac{\partial L(\theta)}{\partial b_{12}}$$

bias involved in layer 1 connected to neuron on layer 2.

$$w'_{111} = w_{111} - \eta \Delta w_{111}$$

•) Similarly for the final outputs and the bias as well.

•) As stated earlier, $\theta$ is the vector of the parameters, which earlier were w, and b. the derivative of loss wrt $\theta$, were the vector containing the partial derivatives of loss wrt the specific parameters of the vector.

↓

The same argument can be extended here. wherein the parameters are segmented layer wise and compose a 2D matrix.

↓

but $\Delta\theta$, still remains the same : vector of the partial derivatives of the constituent elements.

•) An example of derivatives.

ex1. $e^{x^2}$

$$x \rightarrow (sq) \rightarrow (exp) \rightarrow (exp) \rightarrow$$

$h_1 = x^2$ $\quad h_2 = e^{x^2}$ $\quad y = e^{e^{x^2}}$

$$\frac{dy}{dx} = \frac{dy}{dh_2} \cdot \frac{dh_2}{dh_1} \cdot \frac{dh_1}{dx}$$

(intuition for autodifferentiation)

chain rule

Now, $y = e^{h_2}$

$$\frac{dy}{h_2} = e^{h_2}$$

$\Rightarrow h_2 = e^{h_1}$

$$\frac{dh_2}{dh_1} = e^{h_1}$$

$\Rightarrow h_1 = x^2$

$$\frac{dh_1}{dx} = 2x$$

$$\frac{dy}{dx} = \frac{dy}{dh_2} \cdot \frac{dh_2}{dh_1} \cdot \frac{dh_1}{dx}$$

$$= e^{h_2} \cdot e^{h_1} \cdot 2x$$

$\downarrow$ substituting values of $h_1$ and $h_2$

$$= e^{e^{x^2}} \cdot e^{x^2} \cdot 2x$$

ex 2. $\frac{1}{e^{x^2}}$

$$x \longrightarrow \text{sq} \xrightarrow{h_1 = x^2} \text{ex} \xrightarrow{h_2 = e^{x^2}} \text{in} \longrightarrow y = \frac{1}{e^{x^2}}$$

$$\frac{dy}{dx} = \frac{dy}{dh_2} \cdot \frac{dh_2}{dh_1} \cdot \frac{dh_1}{dx}$$

$y = \frac{1}{h_2} \Rightarrow \frac{dy}{dh_2} = \frac{-1}{(h_2)^2}$

$h_2 = e^{h_1} \Rightarrow \frac{dh_2}{dh_1} = e^{h_1}$

$h_1 = x^2 \Rightarrow \frac{dh_1}{dx} = 2x$

$\downarrow$ substituting values

$$\frac{dy}{dh_2} = -\frac{1}{(e^{x^2})^2}$$

$$\frac{dy_{h_2}}{dh_1} = e^{x^2}$$

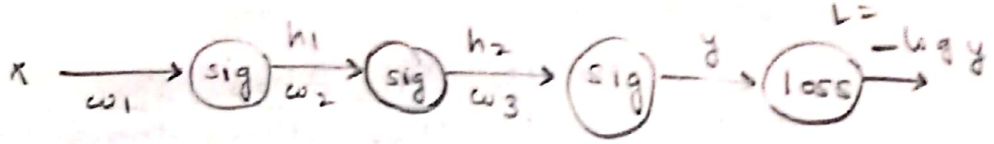
$$\frac{dh_1}{dx} = 2x$$

$\downarrow$

$$\frac{dy}{dx} = -\frac{2x}{e^{x^2}}$$

Scanned by CamScanner

•) The chain rule based computation can be visualised as the reverse of the computation done in forward direction.




$$L = -\log y$$

$$\frac{dL}{dx} = \frac{dL}{dy} \cdot \frac{dy}{dh_2} \cdot \frac{dh_2}{dh_1} \cdot \frac{dh_1}{dx}$$

$$\frac{dL}{dw_1} = \frac{dL}{dy} \cdot \frac{dy}{dh_2} \cdot \frac{dh_2}{dh_1} \cdot \frac{dh_1}{dw_1}$$

$$\frac{dL}{dw_2} = \frac{dL}{dy} \cdot \frac{dy}{dh_2} \cdot \frac{dh_2}{dw_2}$$

$$h_1 = \frac{1}{1+e^{-w_1 x}}$$

$$h_2 = \frac{1}{1+e^{-w_2 h_1}}$$

$$y = \frac{1}{1+e^{-w_3 h_2}}$$

$$L = -\log y$$

↓

So we keep propagating till we reach the variable that is directly dependent on the differentiating variable.

↓

This is a very simple network, but the basic concept remains the same for branched networks as well, and generalise really well.

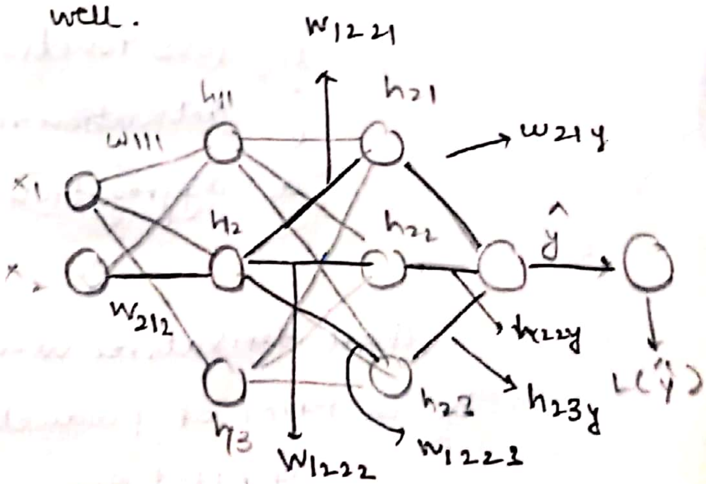
*) Applying chain rule across multiple paths

↓

chain rule is applied across each path separately and then all derivatives are added to get the complete derivative term

↓

since the weight that decides the value of one neuron, might affect other neurons in deeper layers indirectly using the same neuron. i.e. the final loss might be impacted by one weight in multiple ways



$$\frac{\partial L}{\partial w_{212}} = \frac{dL}{dy} \cdot \frac{dy}{dh_{22}} \cdot \frac{dh_{22}}{dh_{12}} \cdot \frac{dh_{12}}{dw_{212}} + \frac{dL}{dy} \cdot \frac{dy}{dh_{21}} \cdot \frac{dh_{21}}{dh_{12}} \cdot \frac{dh_{12}}{dw_{212}}$$

$$+ \frac{dL}{dy} \cdot \frac{dy}{dh_{22}} \cdot \frac{dh_{23}}{dh_{12}} \cdot \frac{dh_{12}}{dw_{212}}$$

↓ partial derivatives

↓

Now there were 3 paths wherein $w_{212}$ was encountered

↓

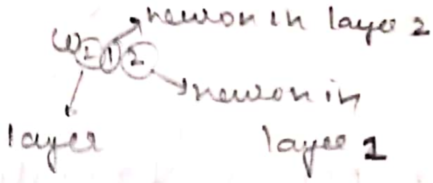
The chain rule concept involves all paths wherein $w_{212}$ or any weight was encountered to operate upon.

←

$h_{ij}$ here are the intermediate o/ps here which themselves contribute to the i/p of the next layer neurons which are then activated.

↓

These are themselves a composition of two ~~variable~~ functions, but that should be pretty trivial.




↓

$$\frac{\partial L}{\partial w_{212}} = \frac{\partial L}{\partial \hat{y}_1} \cdot \frac{\partial \hat{y}_1}{\partial a_{21}} \cdot \frac{\partial a_{21}}{\partial w_{212}}$$

$$\hat{y}_1 = \frac{1}{1+e^{-a_{21}}}$$

$$a_{21} = \sum w_{ij} x_{ij}$$

L : loss function

$\hat{y}$ : activation function

a : aggregation function

↓

These derivatives when computed come out to be in terms of products of elementary values computed during forward propagation

- Building in the dependency order or paths might be more intuitive & safe in some scenarios.

↓ Just for intuition

$$\frac{\partial L}{\partial w_{131}} = \frac{\partial L}{\partial \hat{y}_1} \cdot \frac{\partial \hat{y}_1}{\partial a_{21}} \cdot \frac{\partial a_{21}}{\partial h_{13}} \cdot \frac{\partial h_{13}}{\partial a_{13}} \cdot \frac{\partial a_{13}}{\partial w_{131}}$$

$$+ \frac{\partial L}{\partial \hat{y}_2} \cdot \frac{\partial \hat{y}_2}{\partial a_{22}} \cdot \frac{\partial a_{22}}{\partial h_{13}} \cdot \frac{\partial h_{13}}{\partial a_{13}} \cdot \frac{\partial a_{13}}{\partial w_{131}}$$

↓

So the concept mainly is that the derivative for any weight gets broken into a chain of elementary computations.