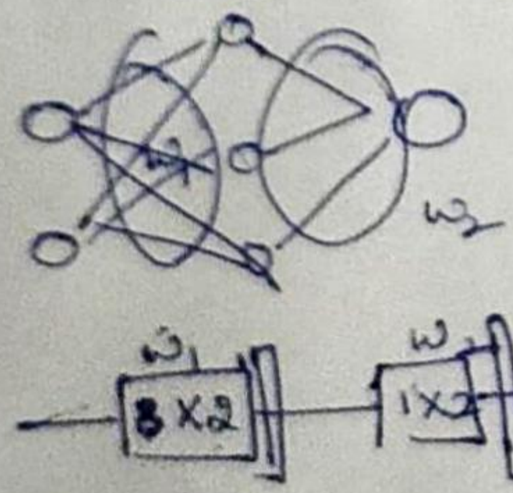
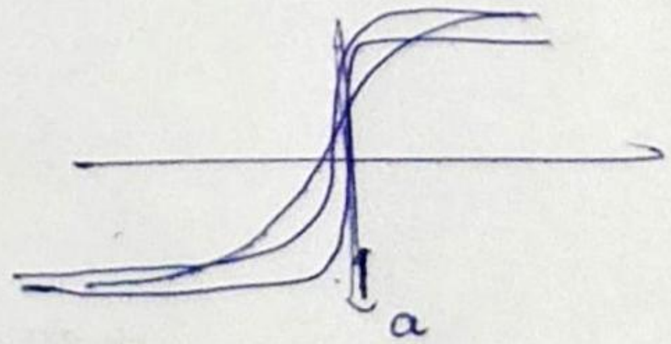
22/10/18 Lec-19

Backpropogation:-



In NLP TB, they generally draw BP's with index



$\phi(x) = \begin{cases} 1 & x \geq 0 \\ -1 & x < 0 \end{cases}$

We are more interested in W_1 and W_2

And what if the ϕ fn has a learnable parameter

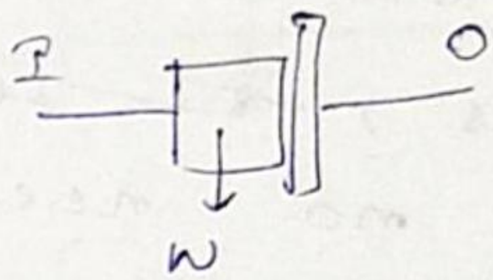
Bias } can be incorporated ↔ Learnable Parameter

Each layer can be ~~learned~~ repr as one matrix and a fn, with no params

If $\frac{\partial L}{\partial O}$ is available and $O \rightarrow$ output

$$\frac{\partial L}{\partial I} = \frac{\partial L}{\partial O} \times \boxed{\frac{\partial O}{\partial I}}$$

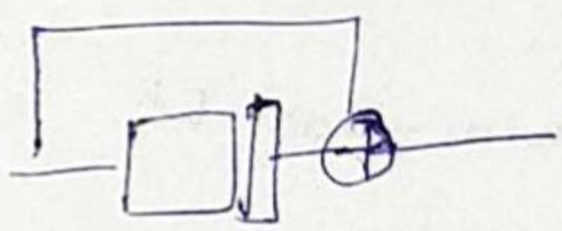
↓ Needed

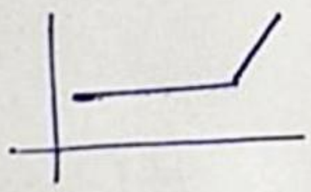


$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial O} \times \boxed{\frac{\partial O}{\partial w}} \rightarrow \text{for every block}$$

$$w^{k+1} \leftarrow w^k - \eta \frac{\partial L}{\partial w}$$

When skip connections are there

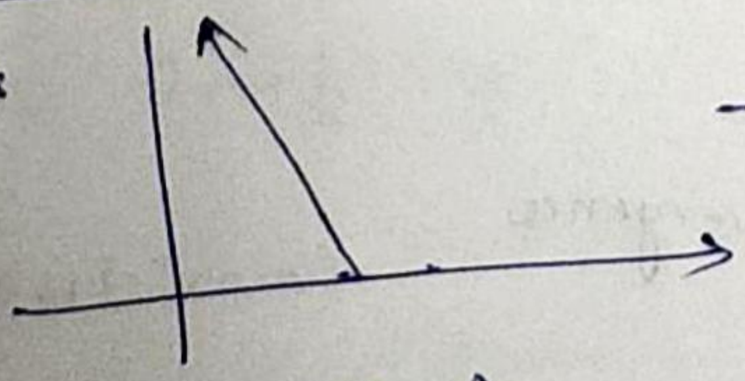


Re-LU 

Easy $\frac{\partial}{\partial x}$ ~~and~~ and non-linearity

- Differentiability at point
- Subgradients are used to solve.
- We ~~know~~ want slopes at a point, ~~but~~ and not the equation $\left(\frac{\partial L}{\partial w}\Big|_{w=w_n}\right)$ so.

Sigmoid
Tanh
were
stopped using

Hinge Loss 

→ A loss that enforces margin ?

$\max(0, (1 - t)\cdot 0)$

Stochastic Version:-

- If loss is same for any random 100 samples, ~~it is same~~ as total 10000 samples, then, no need of using all samples.

~~So~~, That's why stochastic

- "batch" ⟶ Send all → find loss, → find Gradient
  "single sample" ⟶ 1 Forward pass
  Get Loss
  Update
- "mini-batch"

$$\frac{\partial L}{\partial w} = \frac{\partial \sum l_i}{\partial w} \quad \text{Individual samples} \quad = \sum \frac{\partial l_i}{\partial w}$$

So every samples $\frac{\partial}{\partial x}$ is ~~added~~ calculated

If 100 give same $\frac{\partial}{\partial x}$ ~~and~~ as 1000 ~~give~~ $\frac{\partial}{\partial x}$

Then 100 is enough

---

① Initialize
② Forward pass → find loss
③ Back prop → find w
④ Repeat 2-4 until convergence

Converges:

$L \geq 0$

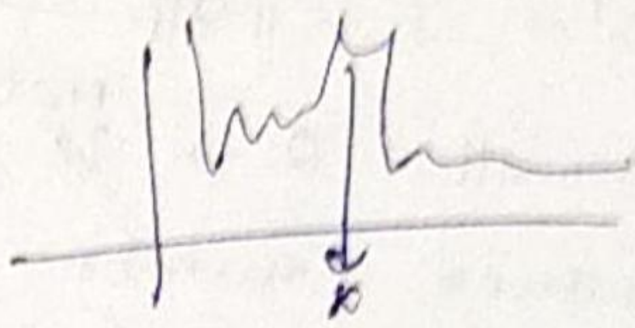
$L^0 \to L^1 \to L^2 \cdots L^{n-1} - L^w$

$L^0 < L^1 < L^2 \cdots$

Unless u are a shit head, Q: Loss ~~should~~ would never converge.

. will we reach correct minima?

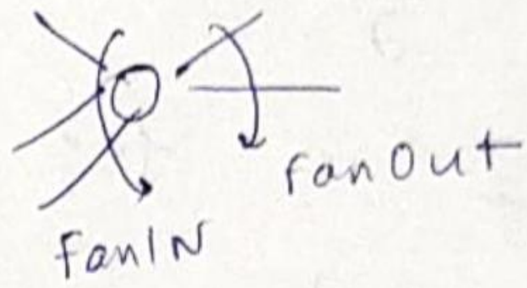
A) Don't be worried about getting stuck at *, coz, u are seeing 1D but other directions w, bz may not be minima → (saddle point)



Initialization

1) Random (why not zero)? [0 to 1] or [-1 to 1]

. Xaviers, He's Init ||

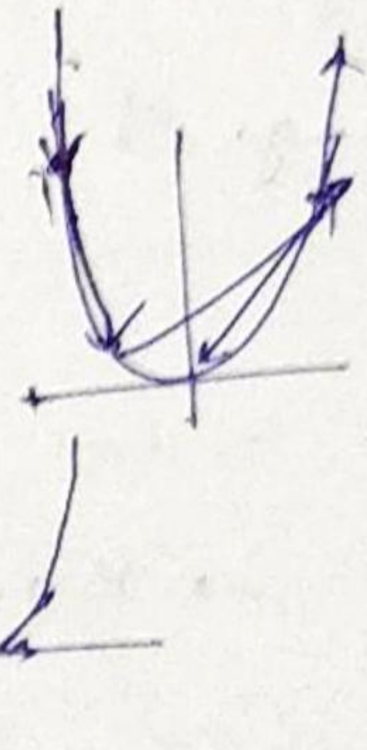


Update rule

1) Momentum $\theta^{k+1} \leftarrow \theta^{k} - \eta \frac{\partial L}{\partial \theta}$

- Previous direction

— Nestrerov momentum



Adaptive Gradient

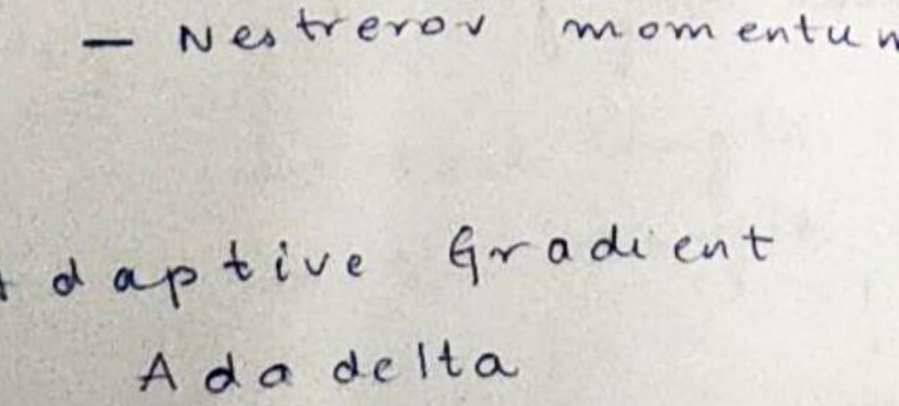
Ada delta

Ada del

RMs Prop

Adam

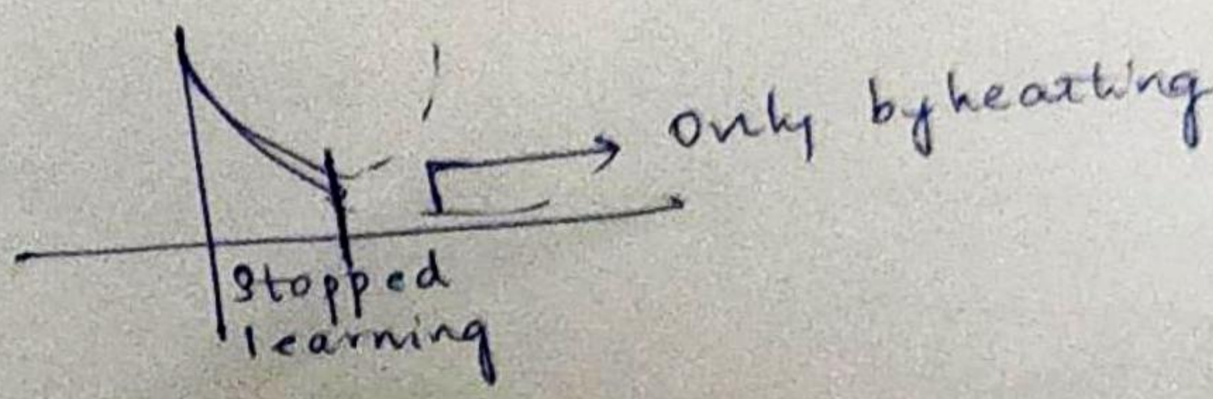
Termination: Overfitting



It should start at the same point

Many samples → that it is superspecializing for training data.

So termination



iii) $J' = J + \|\theta\| \longrightarrow L_0, L_1, L_2$

If small $\theta \rightarrow$ ~~If~~ Not enough weights : ~~Some~~ No by-heart

So, reduce, number of params.

Weight decay

- If small weights, remove those edges

force to recover by back propogation

$$\sqrt[n]{x_1^n + x_2^n + \dots}^{1/n}$$

$$\frac{1}{(x_1 + x_2 + \dots)^{\frac{1}{n}}} \times n x_1^{n-1}$$

$$\frac{\partial J'}{\partial -} = \frac{\partial J}{\partial -} + \frac{\partial \|\theta\|}{\partial}$$

In practice L_2 is for $\|\theta\|$

[Pruning of networks]

Second order methods:

$H^{-1} \rightarrow$ Bad // $(d \times d)$ $\longrightarrow$ fastu convergence (Good)

Neural networks:

$\phi = 2x$

↳ Linear fn

Then it will ~~combine~~ collapse into 1

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

$$\frac{\partial E}{\partial w_2} = \dots (y - \hat{y}) \frac{\partial y}{\partial w_2}$$

$$y = \phi(w^T x_i)$$