Introduction to Neural Nets

→ (Deep-Learning Module)

↳ ~10-11 classes

Masters (18M)

electives

- CV-DL → ~1 month ✓
- NLP-DL → ~1 month ✓
- MLOps & Big Data → ~1 month
- Adv DSA → ~2½ months

Chrome File Edit View History Bookmarks Profiles Tab Window Help

Career outcomes

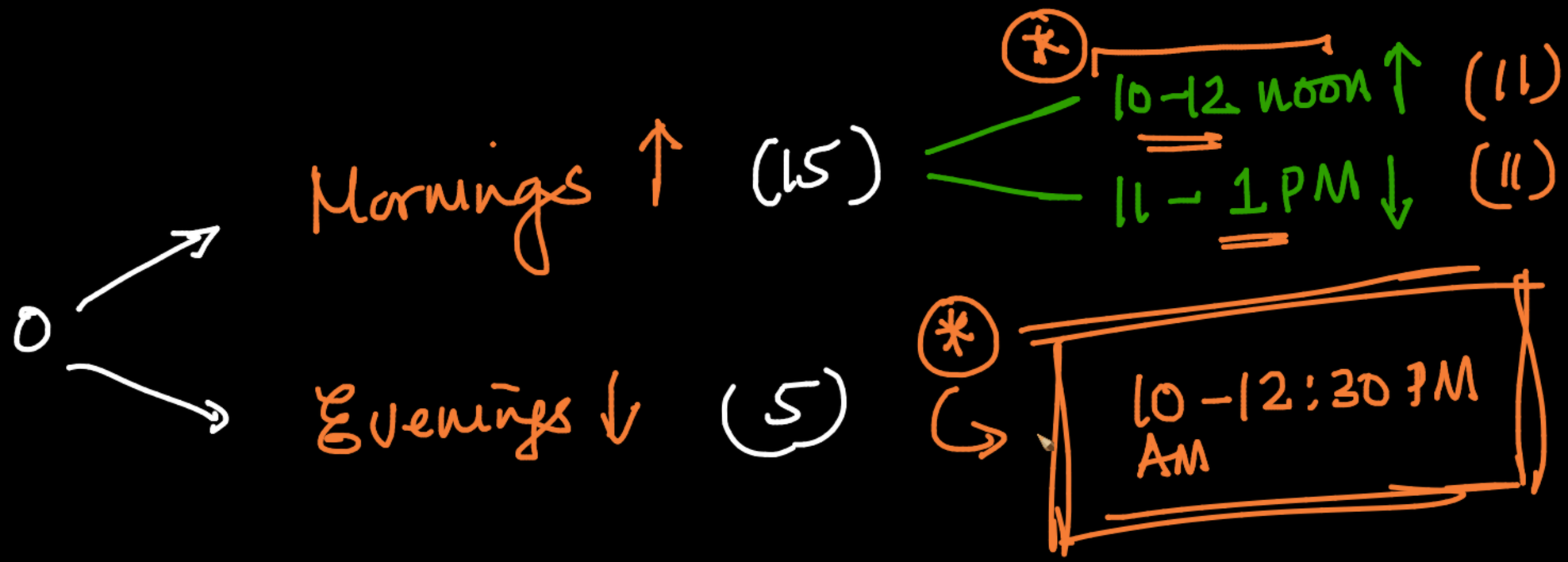
Sat & Sun

(recap + Interview Prep Sessions)

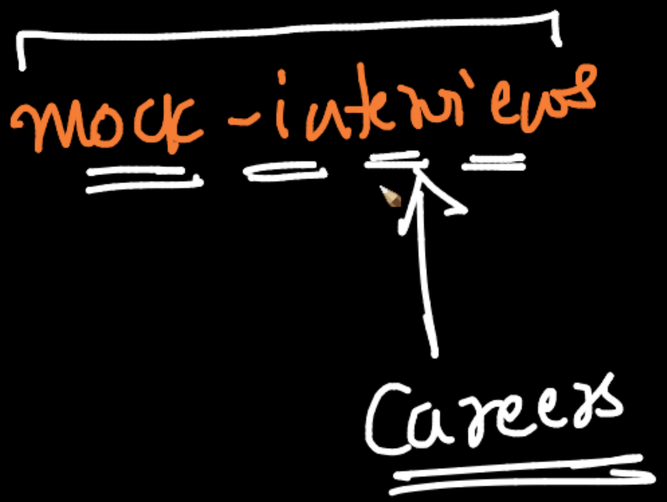
MWF → DL

1. Programming
2. Pods + Stats
3. Math for ML
4. Classification & Regression ML-1
5. ML-2 (rest of Classical ML)
6. SQL

Suggestion: Lots of case-study sessions


Mornings ↑ (15)
10-12 noon ↑ (11)
11 - 1 PM ↓ (11)
Evenings ↓ (5)
10-12:30 PM
AM

(*) Please complete



{ Neural Networks (NN)

Practice

Some DL-NN

Tabular Data → GBDT / RF ...

(2006 - ...)
(2012 - ...)

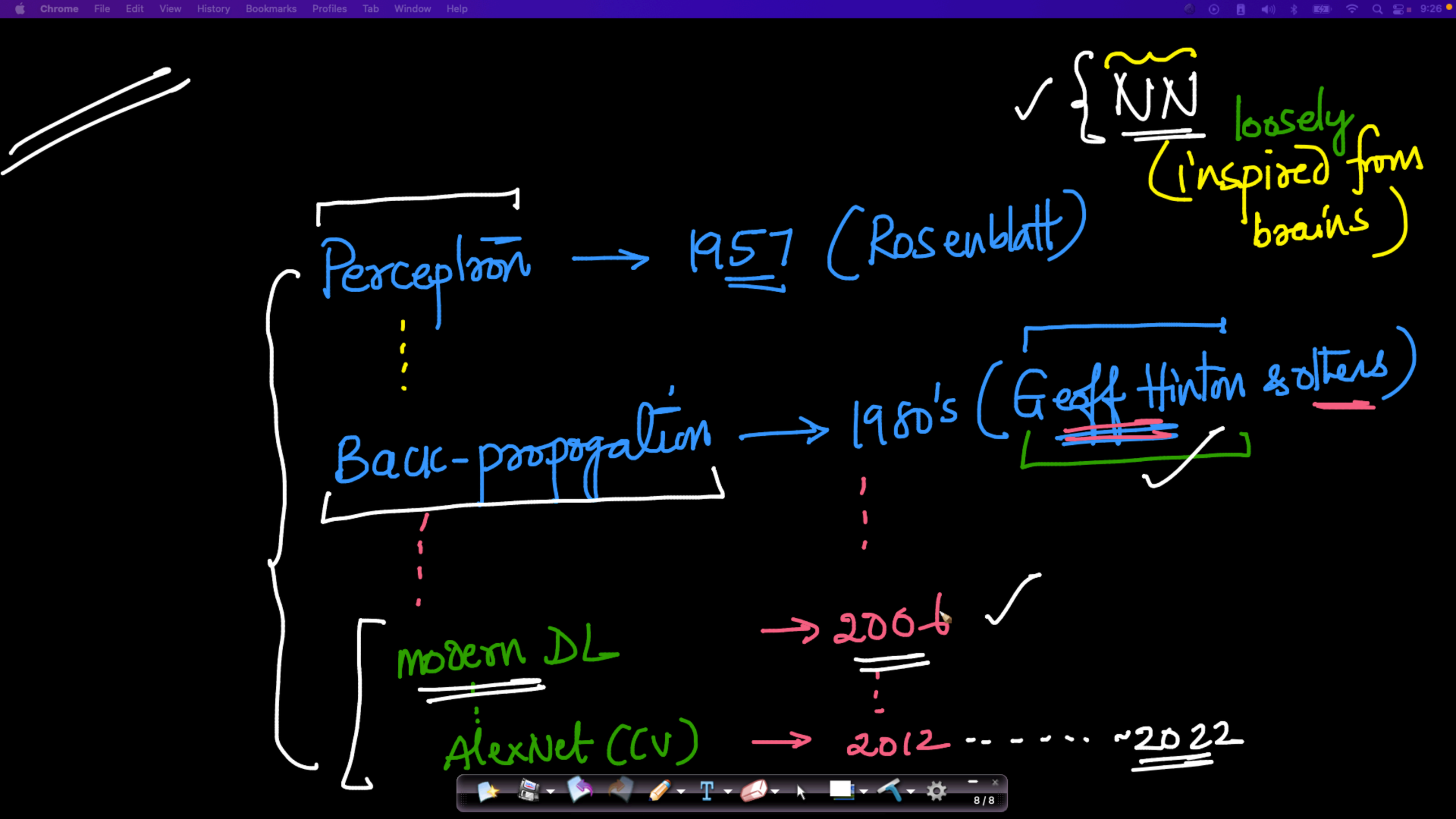
{ Image / Speech / Complex Time Series / Text } → DL (NN)

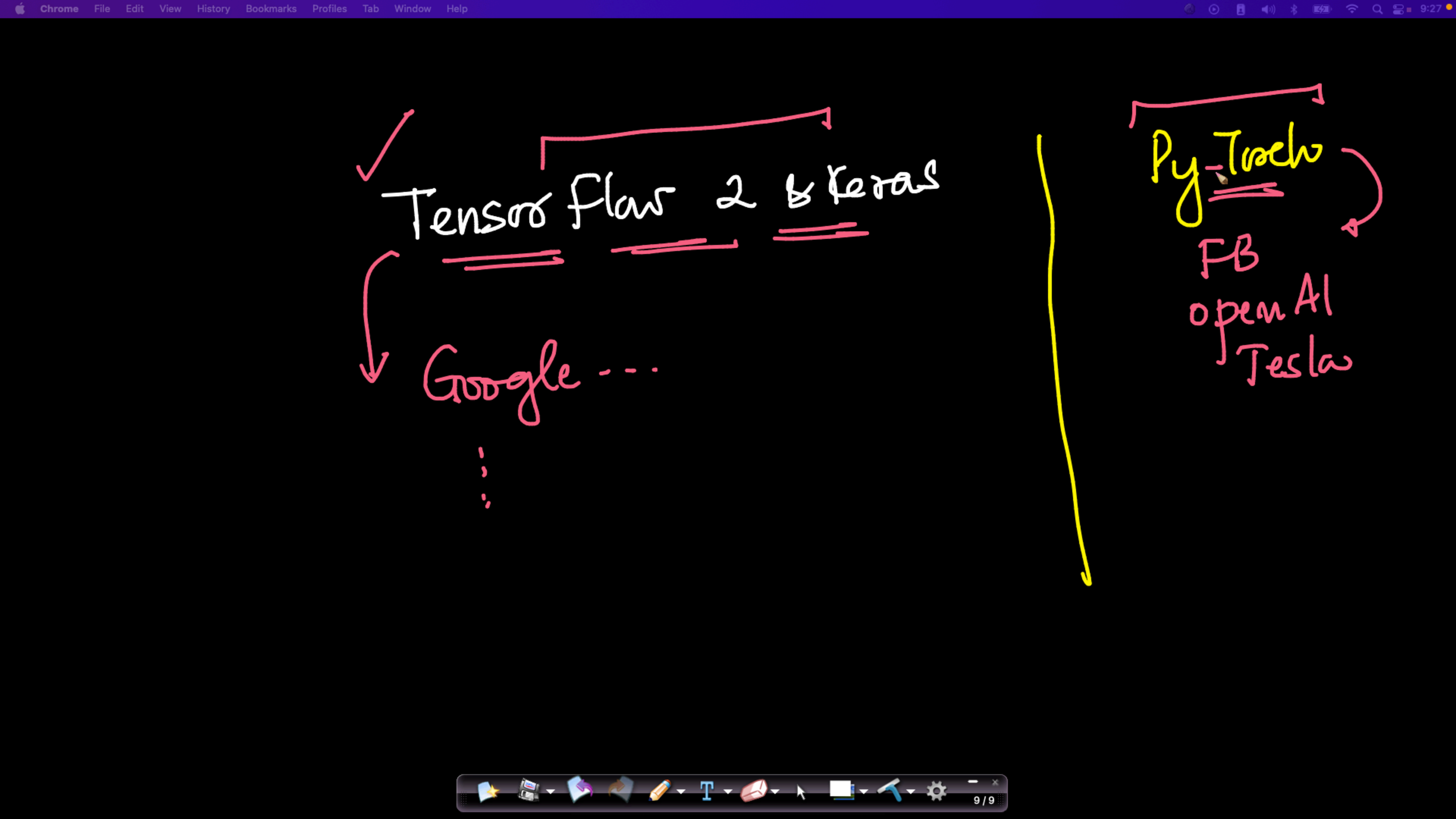
RecSys (NCF)

[1000's of research papers]

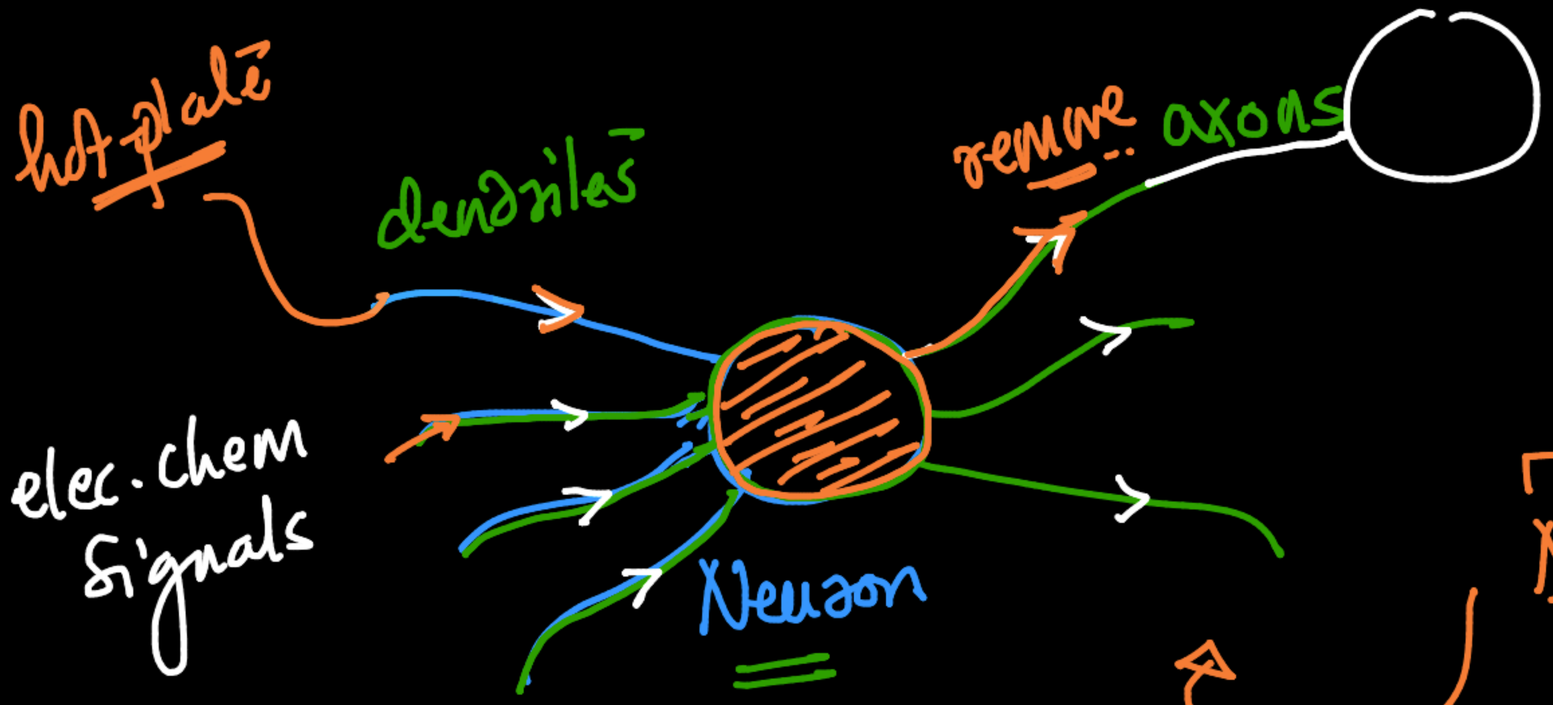
Obj: equip you with strong foundations → any new model
+
cover most widely used models
+
research papers

Assumption: classical ML + Math (recap)


✓ { NN (inspired from brains) loosely
Perceptron → 1957 (Rosenblatt)
Back-propogation → 1980's (Geoff Hinton & others)
modern DL → 2006 ✓
AlexNet (CV) → 2012 ---- ~2022


Chrome File Edit View History Bookmarks Profiles Tab Window Help
9:27
Tensorflow 2 & Keras
Google ...
Py-Torch
FB
open AI
Tesla
9 / 9

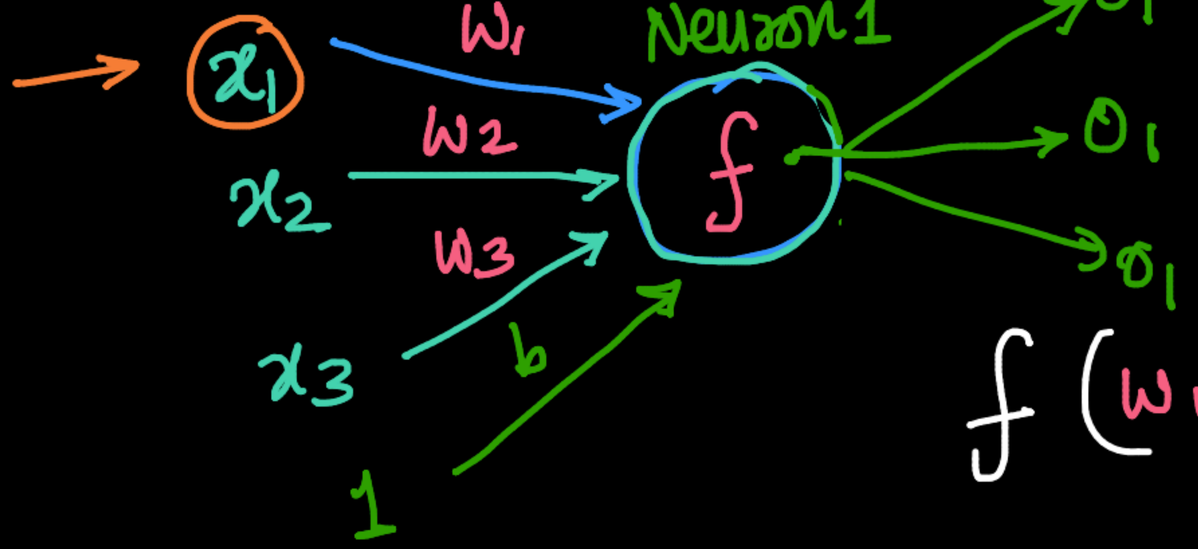
NN: "loosely" inspired by brains



Biological Neuron

Not studying

Artificial Neuron



$$f(w_1x_1 + w_2x_2 + w_3x_3 + b) = o_1$$

f : activation fn

Terminology:

$$O_i = f\left(\sum_{j=1}^{d} x_{ij} w_j\right)$$

(output)

- x_{ij}: inputs
- w_j: weights
- f: activation fn

$x_i \in \mathbb{R}^2$ → O_i

Logistic Regr:

$[x_{i1}\ x_{i2} \cdots x_{id}]^T$

$D_{Tr} = \{(x_i \in \mathbb{R}^d, y_i)_{i=1}^{n}\}$

$\hat{y}_i = \text{sigmoid}(w^T x_i + b)$

w: d-dim

b: scalar

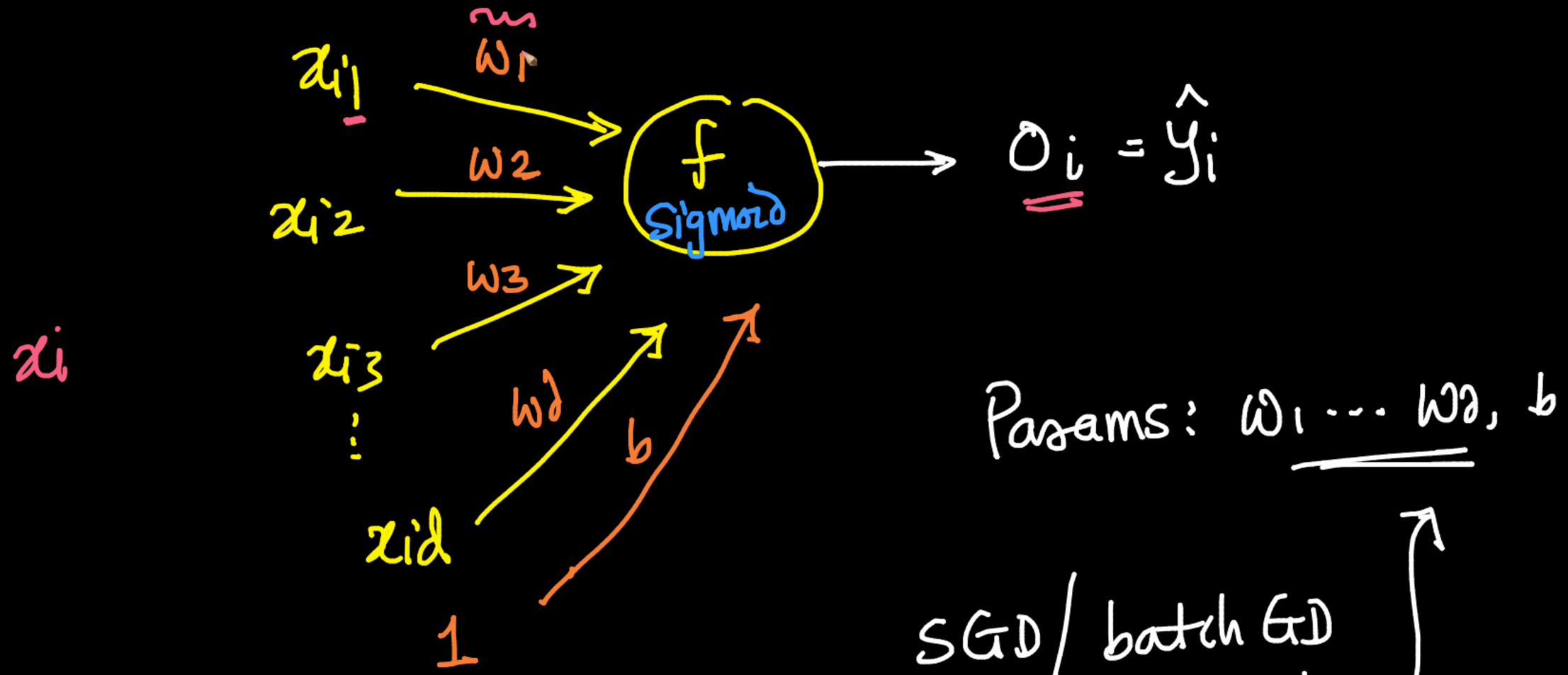
ANN: Perceptron

logistic Reg (NN)

MLP

back prop

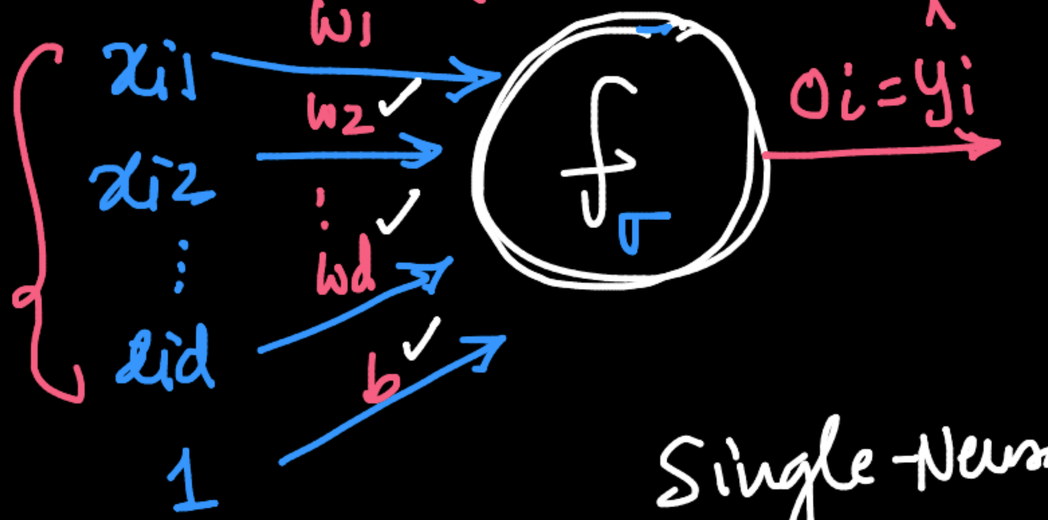
code



Params: $w_1 \cdots w_d, b$

SGD / batch GD

NN-way of representing a Logistic Regr.

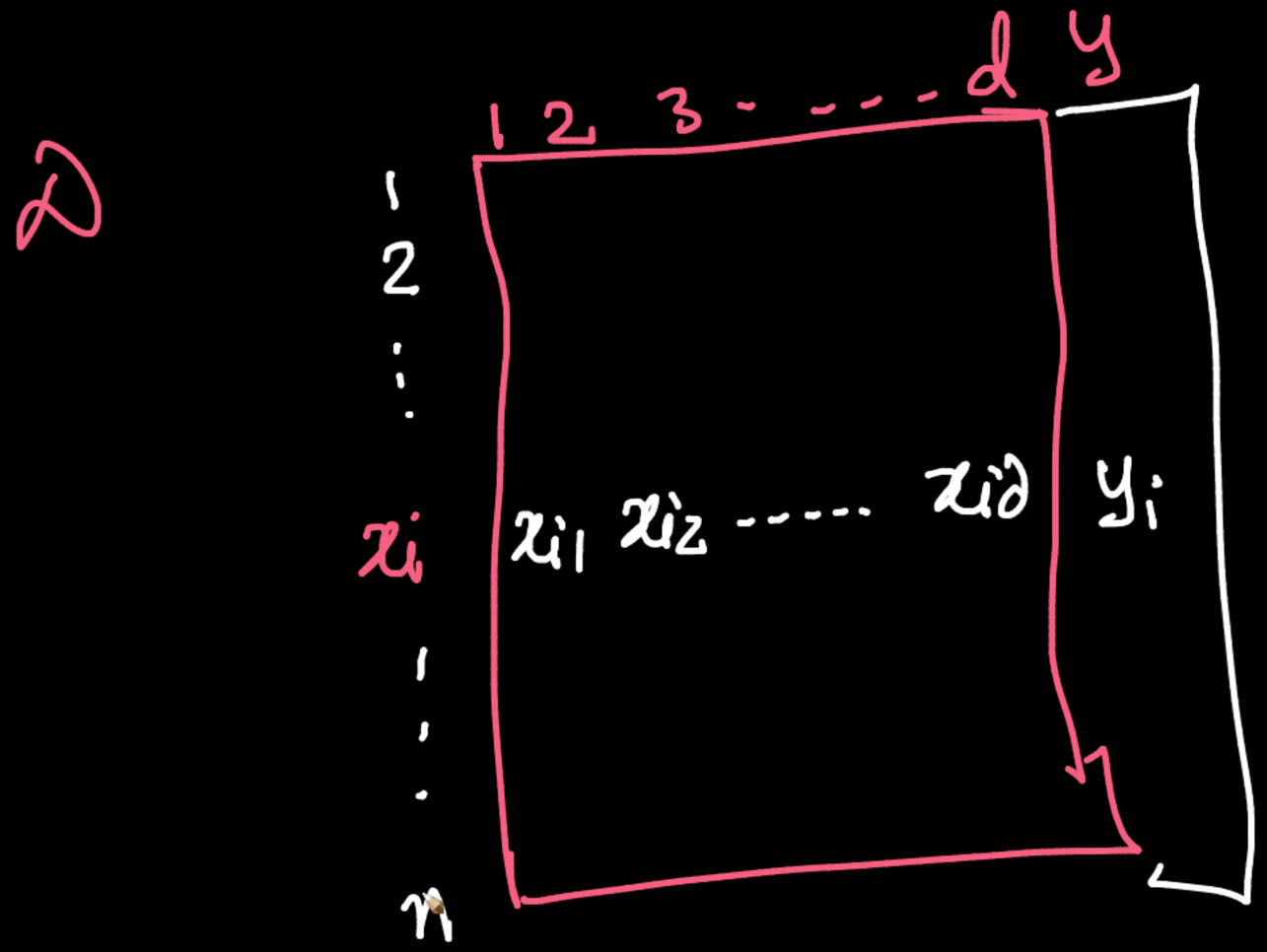


$\forall x_i$

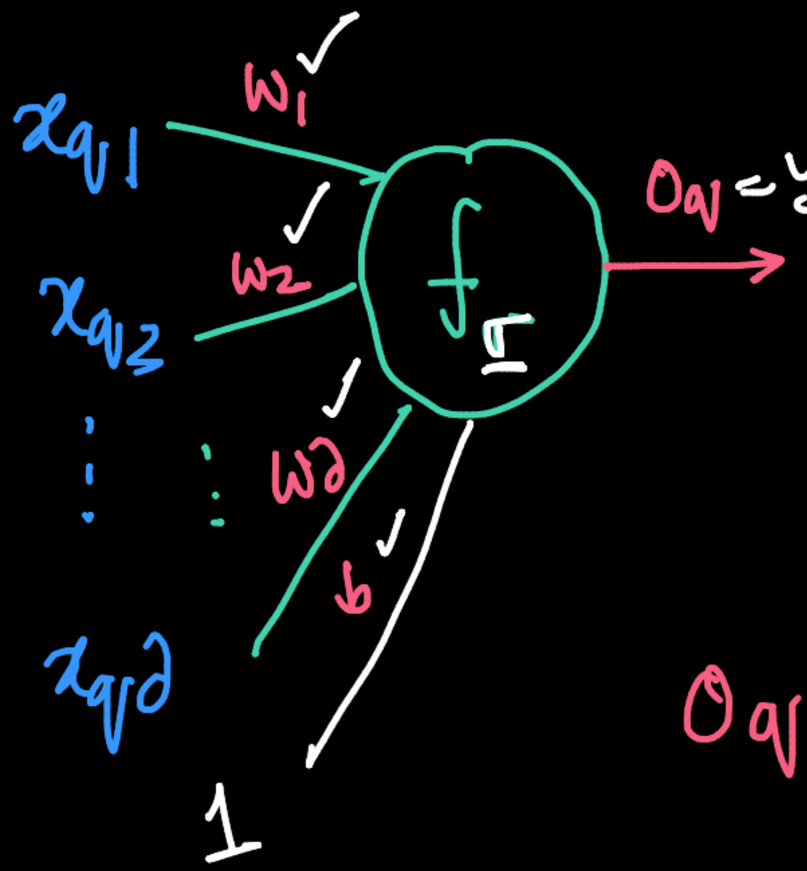
Single Neuron

Forward propagation : $f_\sigma\left(\sum_{j=1}^{d} w_j x_{ij} + b\right)$

$= f_\sigma(w^T x + b)$



$x_q \in \mathbb{R}^d$



$o_q = \hat{y}_q = P(y_q = 1 \mid x_q, w)$

$o_q = \hat{y}_q = \sigma(w^T x_q + b)$

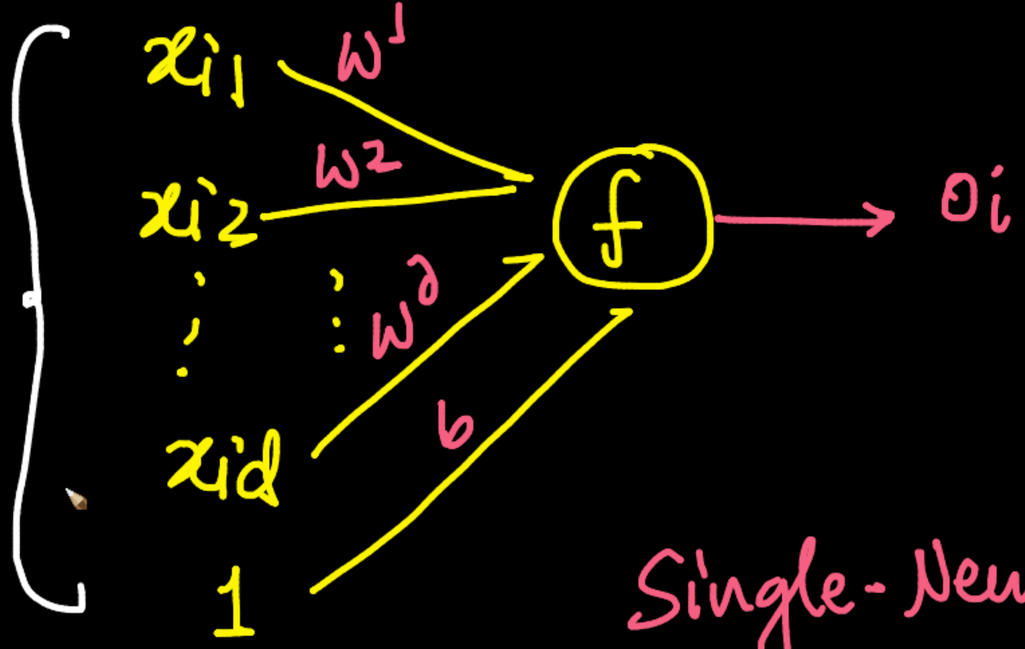
FP: $x_q \rightarrow o_q = \hat{y}_q$

Perceptron

$x_i \in \mathbb{R}^d, \; y_i \in \{0, 1\}$

$$o_i = f_{\text{perceptron}}(x_i, w, b)$$

$$= \begin{cases} 1 & \text{if } w^T x_i + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

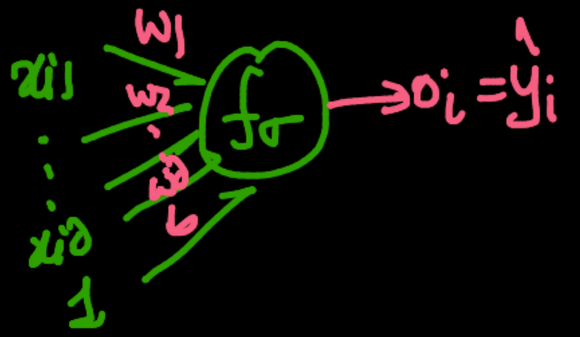


Single-Neuron

Eqns ⟶ Geom ⟶ NN-diagram

Logistic reg :- $\sigma(w^T x_i + b) = \hat{y}_i$

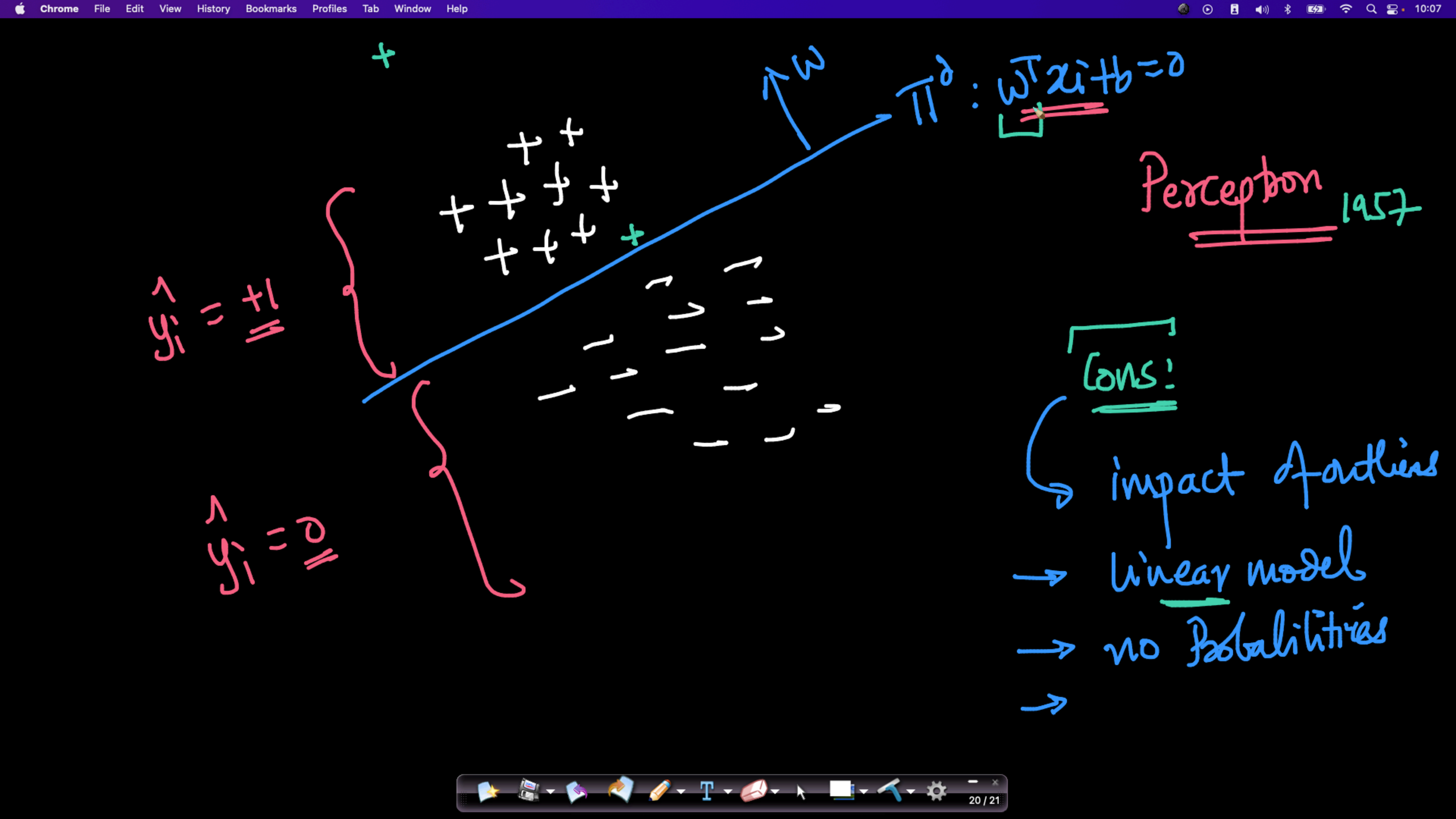
π^d; squashing (σ)



Perceptron :- $f_{perceptron}(w^T x_i + b) = \hat{y}_i$

π^d; no-squashing

$$f(x_i, w, b) = \begin{cases} 1 & ; w^T x_i + b > 0 \\ 0 & \text{o/w} \end{cases}$$


w
$\pi^d : w^T x_i + b = 0$
Perceptron 1957
$\hat{y}_i = +1$
$\hat{y}_i = 0$
Cons!
impact Aarthi
→ linear model
→ no Probabilities
→

Train

Perceptron

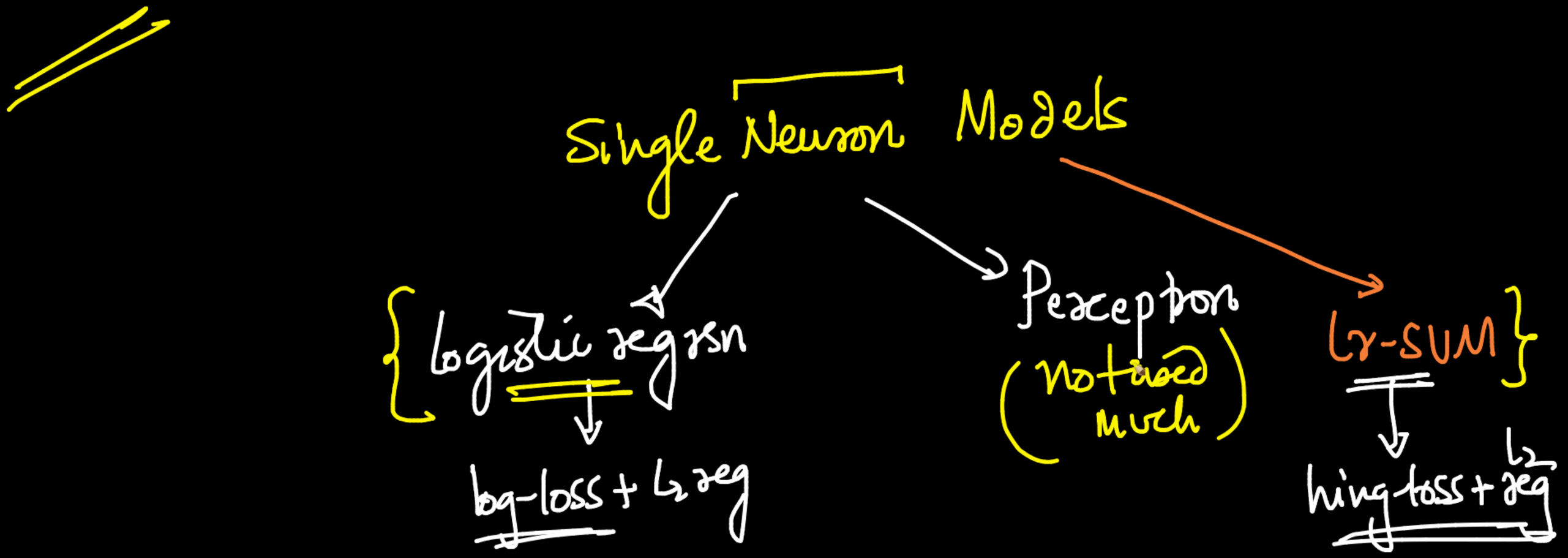
$x_i \longrightarrow y_i \rightarrow 0$ or $\rightarrow 1$

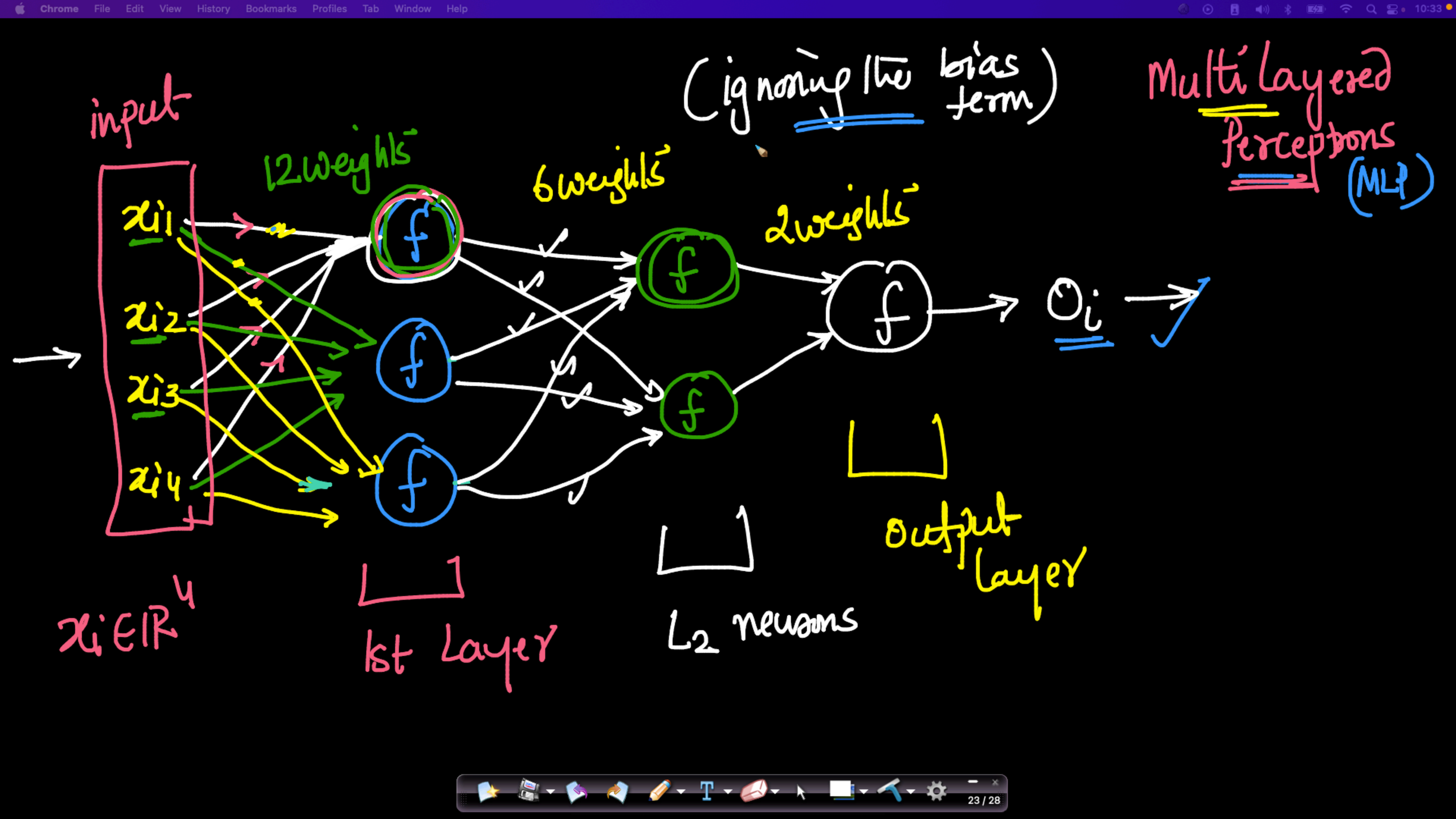
$\hat{y}_i = f_p(w^T x_i + b) \rightarrow 0$ or $\rightarrow 1$

GD

$$\min_{w,b} \sum_{i=1}^{n} \text{Loss}(y_i, \hat{y}_i) + \lambda \, L_2\text{-reg}(w_j)$$

even use MSE


Single Neuron Models
Logistic regrsn
log-loss + L2 reg
Perceptron
(not used much)
L2-SVM
hing-loss + L2 reg


(ignoring the bias term)
Multi Layered Perceptrons (MLP)
input
12 weights
6 weights
2 weights
x_{i1}
x_{i2}
x_{i3}
x_{i4}
f
O_i
$x_i \in \mathbb{R}^4$
1st Layer
L_2 neurons
output layer

Q

Why do MLP?

(Algebra)

fog got

f_1 : +

f_2 : −

f_3 : *

f_4 : /

f_5 : sqrt

f_6 : sin

f_7 : cos

$$f_1(x_{i1}, x_{i2})$$

$$= x_{i1} + x_{i2}$$

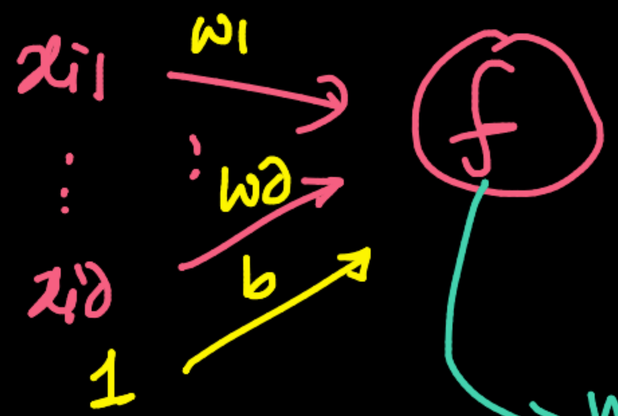
$$f_6(f_1(x_{i1}, x_{i2})) = \sin(x_{i1} + x_{i2})$$

Composition of fn

$$fog(x) = f(g(x))$$

$$gof(x) = g(f(x))$$

Construct Complex fns as gof or fog over & over again.

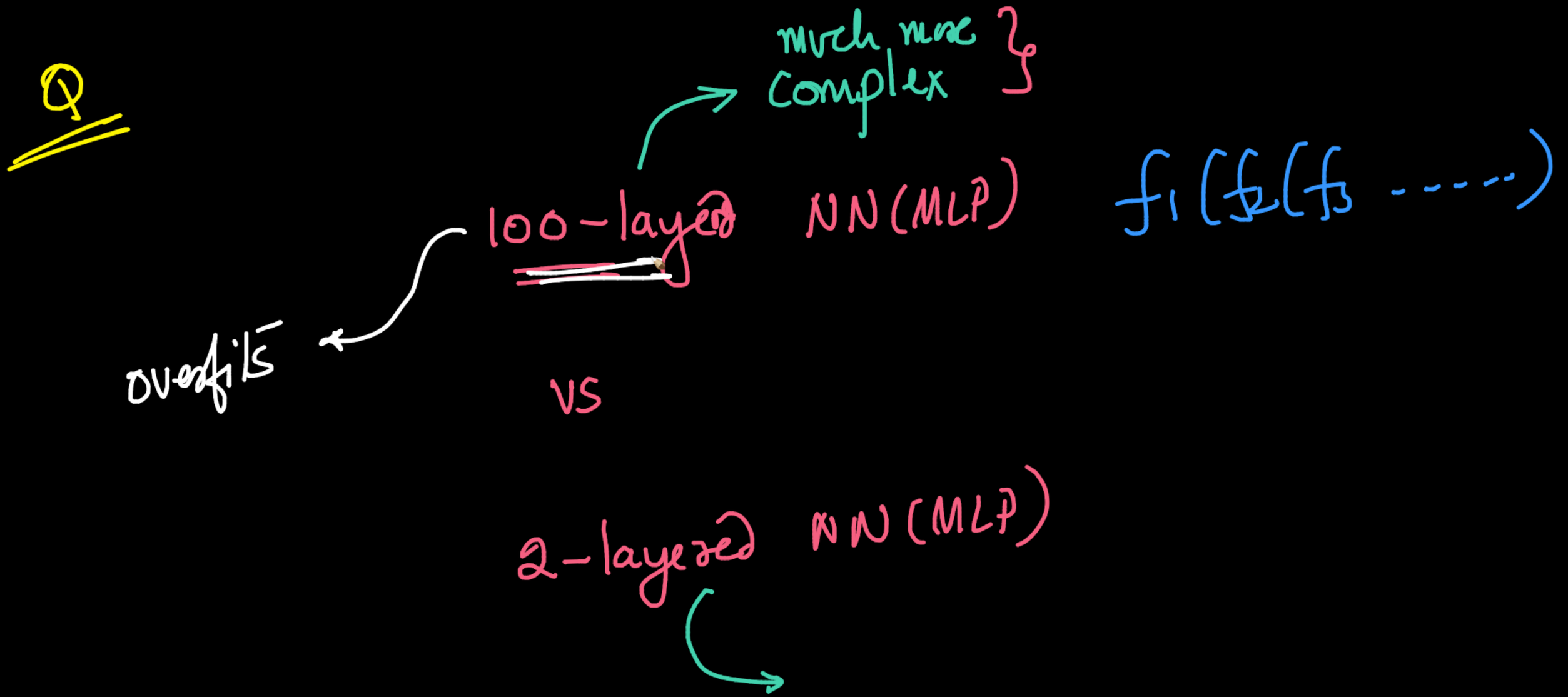


$\underline{f}(w^T \underline{x_i} + b)$

↳ linear fn

non-linear

(always)

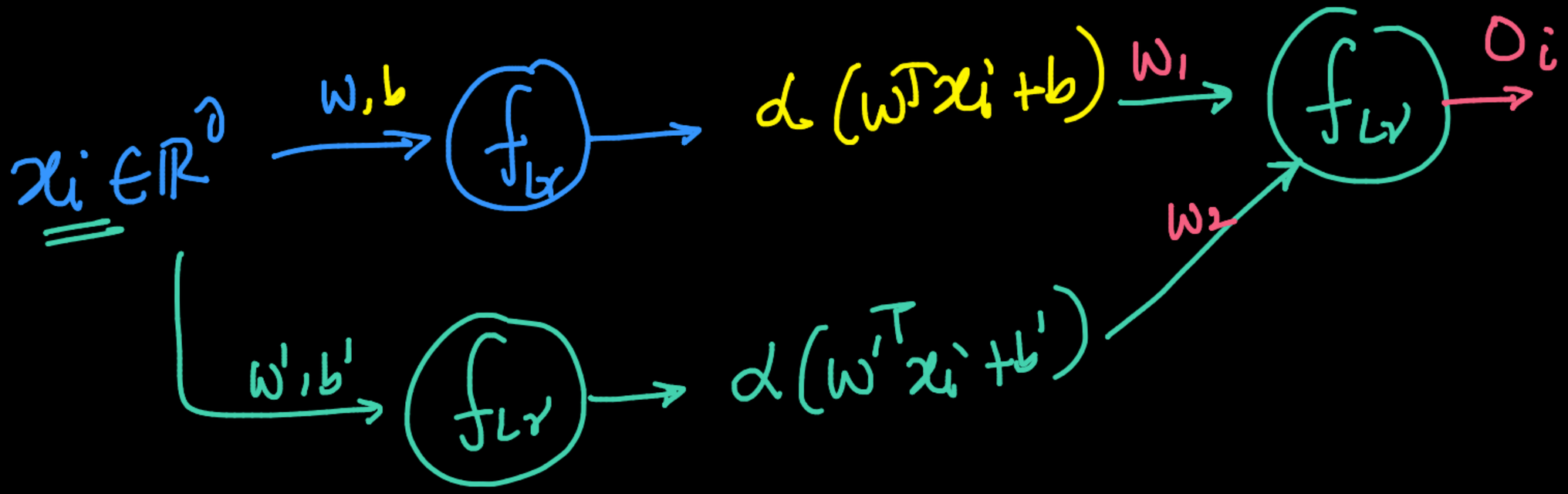

Q
much more complex
100-layered NN (MLP)
overfits
vs
2-layered NN (MLP)
f1(f2(f3 ----)

Q

What happens if each activ. fu in a MLP is a linear-fu.

$$f(x_i, w, b) = f_{LR}(w^T x_i + b) =$$

$$\alpha (w^T x_i + b) + \beta$$



$$O_i = \alpha\left(\alpha w_1 (w^T x_i + b) + \alpha w_2 (w'^T x_i + b')\right)$$

$$= \alpha^2 w_1 (w^T x_i + b) + \alpha^2 w_2 (w'^T x_i + b')$$

If all of our activation fns are linear

then MLP = Linear-model

Hence always use non-lr activation fns ...

in DL & NN

different actvn fns in various layers

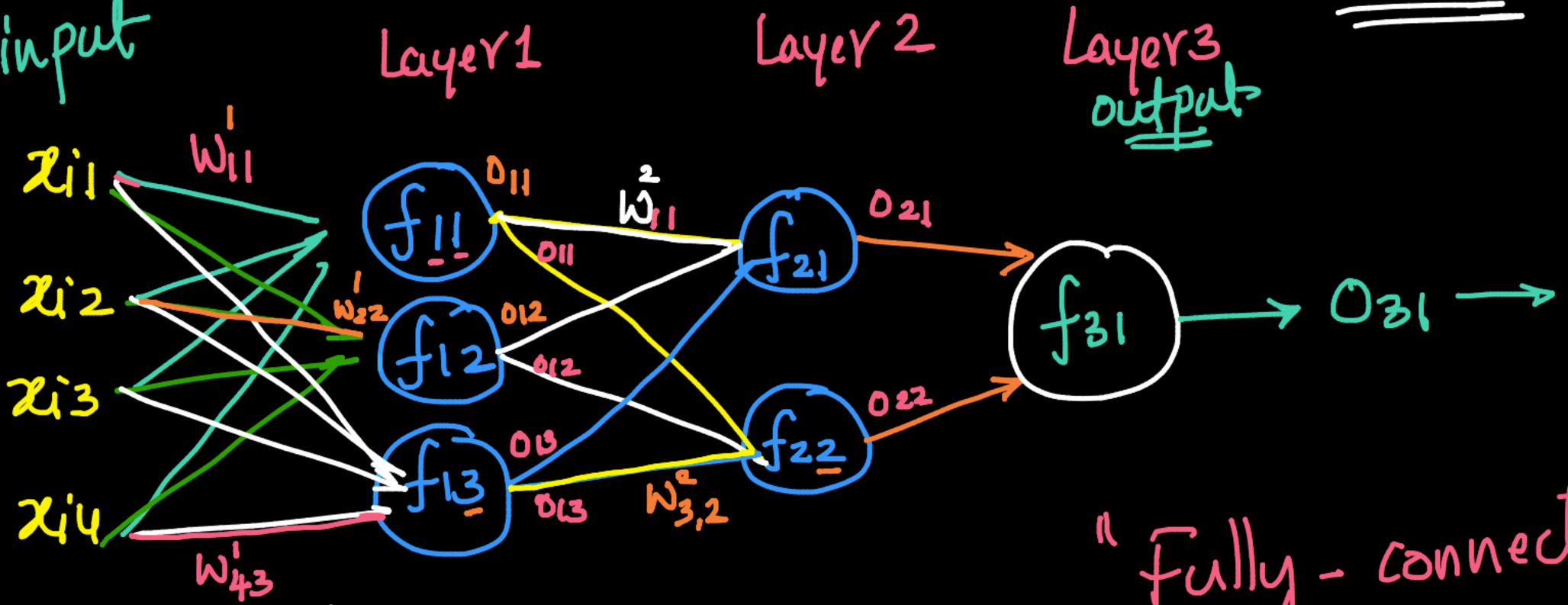
↓

You-can (theoritically)

Practically: not much incremental value

e.g: sigmoid → 1980's & 90's

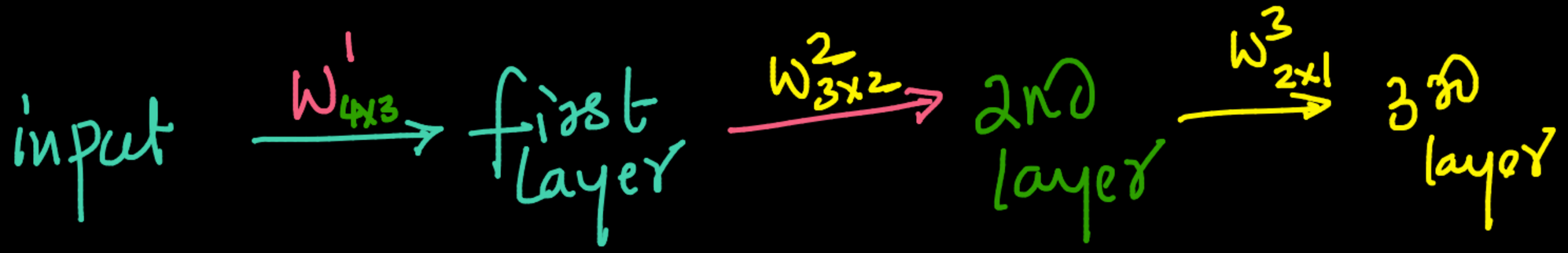
Math



"Fully-connected" MLP

W^k_{ij}: k ← nextLayer; i ← from; j ← to

O_{ij}: Layer; index of neuron

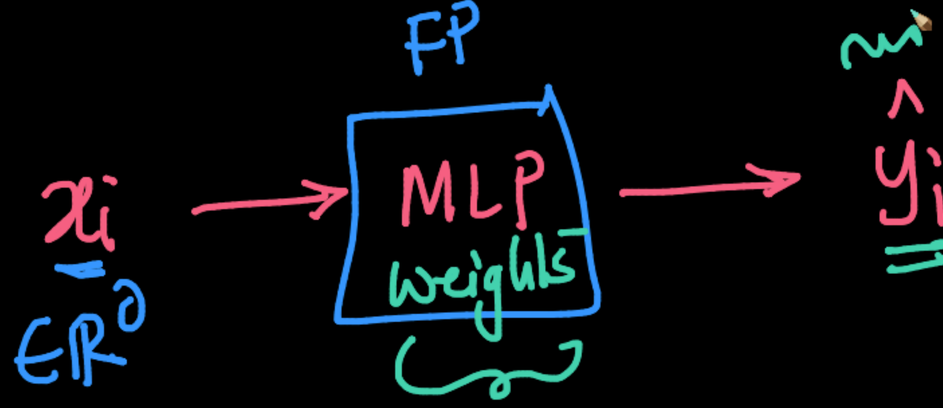


$$W^1 = \begin{bmatrix} w^1_{11} & w^1_{12} & w^1_{13} \\ w^1_{21} & w^1_{22} & w^1_{23} \\ \vdots & & \\ w^1_{41} & w^1_{42} & w^1_{43} \end{bmatrix}_{4\times3}$$

→ Matrices

Classfn: $\mathcal{D}_{Tr} = \{(x_i, y_i)\}_{i=1}^{n}$

Train an MLP



$\text{Loss}(y_i, \hat{y}_i) = \text{loss}_i$

$$\min_{w} \sum_{i=1}^{n} \text{loss}_i$$

Find the best weights

① initialize all weights w randomly

② each x_i → FP → MLP (w) → o_i → $\widetilde{loss_i}$

$$\min_{w} \sum_{i=1}^{n} loss_i$$

Loss ✓

{ all activation fns & loss fn have to be differentiable

Brute force

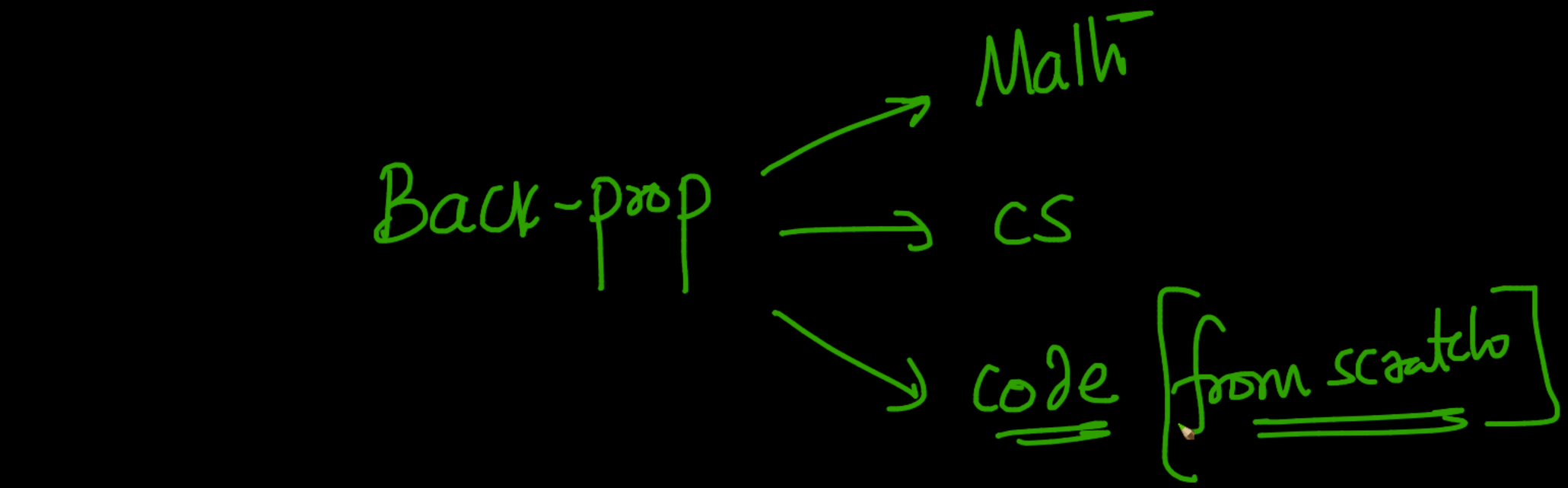
highly Sub-optimal

(3) GD →

$$w_{ij}^{k}\Big|_{new} = w_{ij}^{k}\Big|_{old} - \eta \frac{\partial \widetilde{Loss}}{\partial w_{ij}^{k}}\Bigg|_{w_{ij}^{k}\,old}$$

(k ← next layer)

(4) repeat Step 3 $\forall i, j, k$ till convergence


Back-prop
Math
cs
code
[from scratch]

Q

$$Loss = \sum_{i=1}^{n = 1\,Million} loss_i$$

GD: $\frac{\partial Loss}{\partial w}$ all the n datapts

SGD: $\frac{\partial Loss}{\partial w}$ using one random pt

Most widely used { mini batch - SGD: $\frac{\partial L}{\partial w}$: random batch of k-pts
(k)

Q

Toy-ex!

$$loss = \sum_{i=1}^{n} loss_i$$

$$loss_i = (y_i - \hat{y}_i)^2$$

$$\frac{\partial Loss}{\partial w_1} = \sum_{i=1}^{n} \frac{\partial Loss_i}{\partial w_1}$$

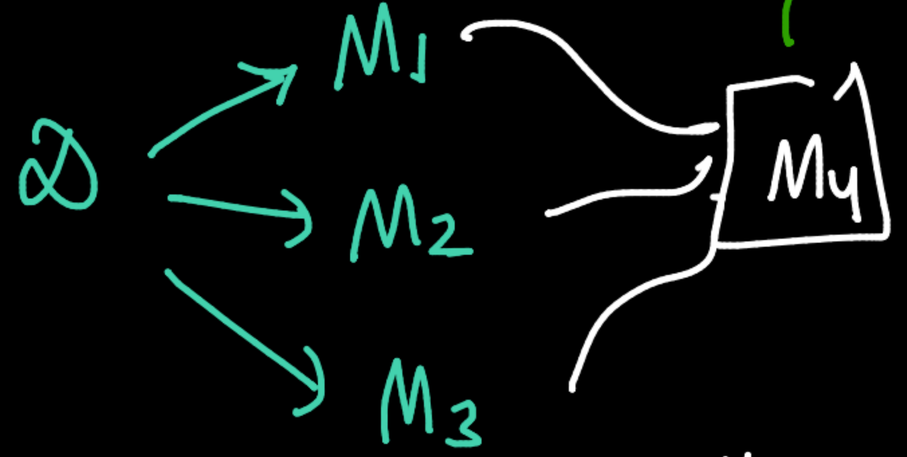
$$\frac{\partial \text{Loss}_i}{\partial w_1} = \frac{\partial}{\partial w_1}\left(y_i - \overbrace{f_2(f_1(x_i w_1) \times w_2)}^{\hat{y}_i} \right)^2$$

f_1, f_2 are differentiable

Chain-rule

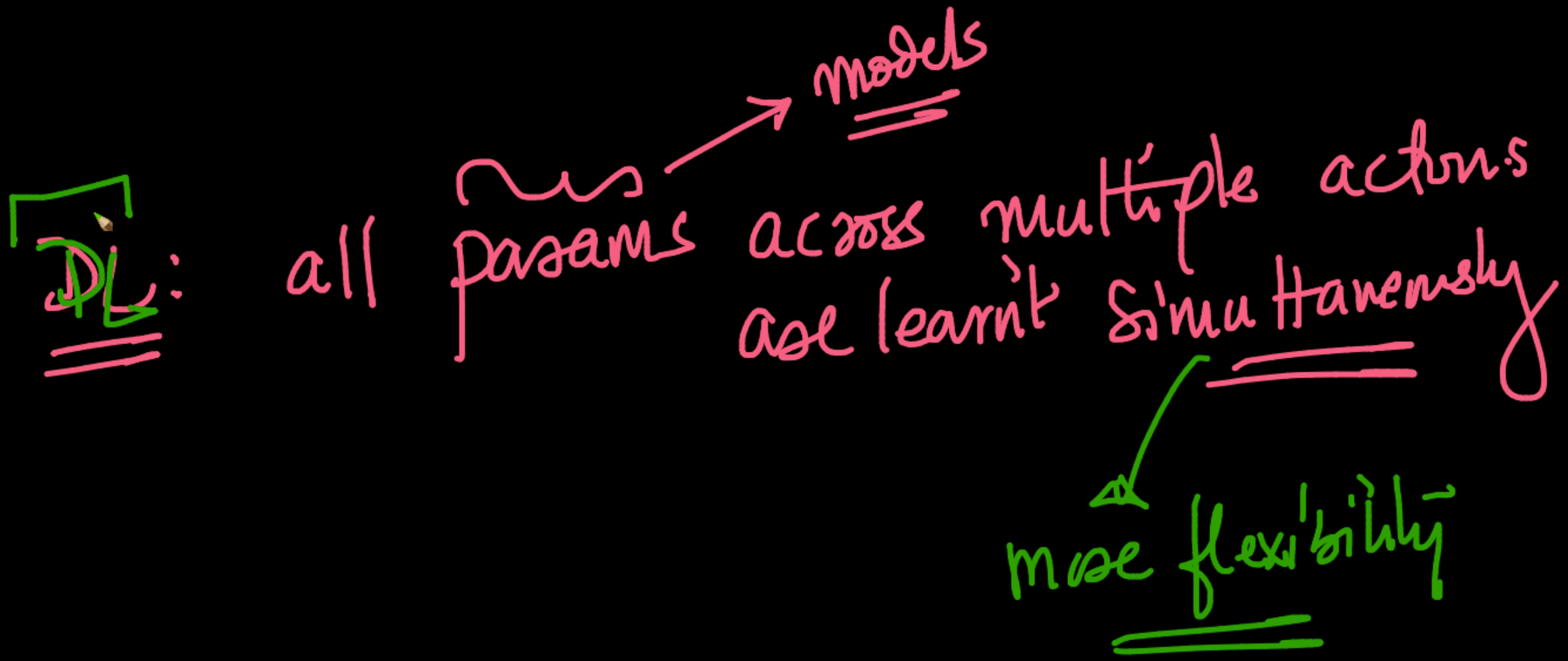
Q

x { Stacking:



→ was built after M1, M2 & M3

→ all models were built indep

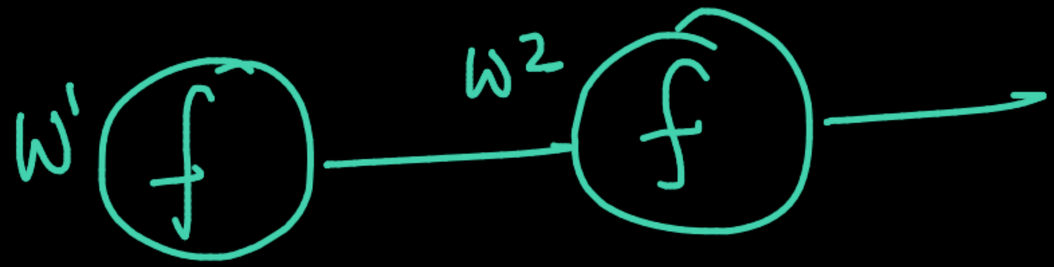


Boosting :

$h_i(x)$ fixed

$h_i(x) \rightarrow h_{i+1}(x) \rightarrow h_{i+2}(x)$

DL:



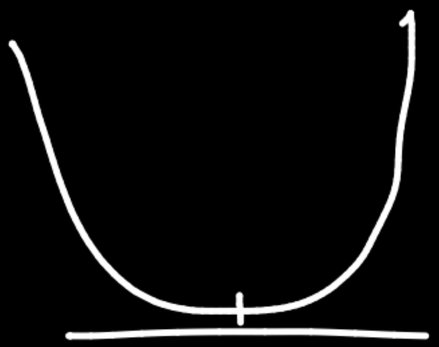
Simultaneously trained

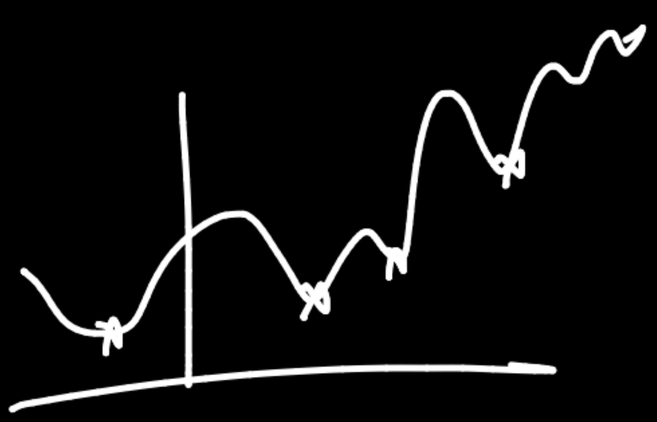
DL

$f(f(f(x_i)))$ with w^3 w^2 w^1 → Very Complex

→ non-convex

Convex!

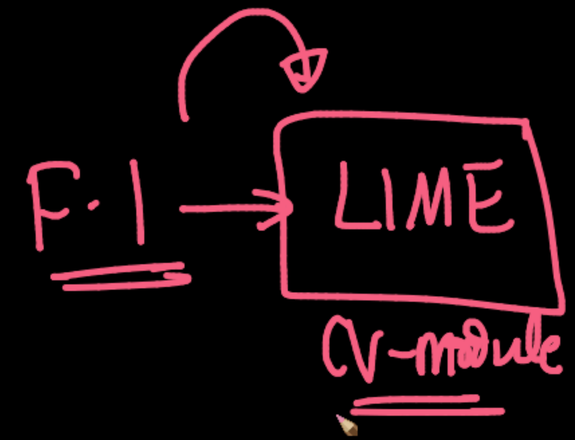




Q

Lr. reg → few sec (1 MM)

DL → Mins / hrs

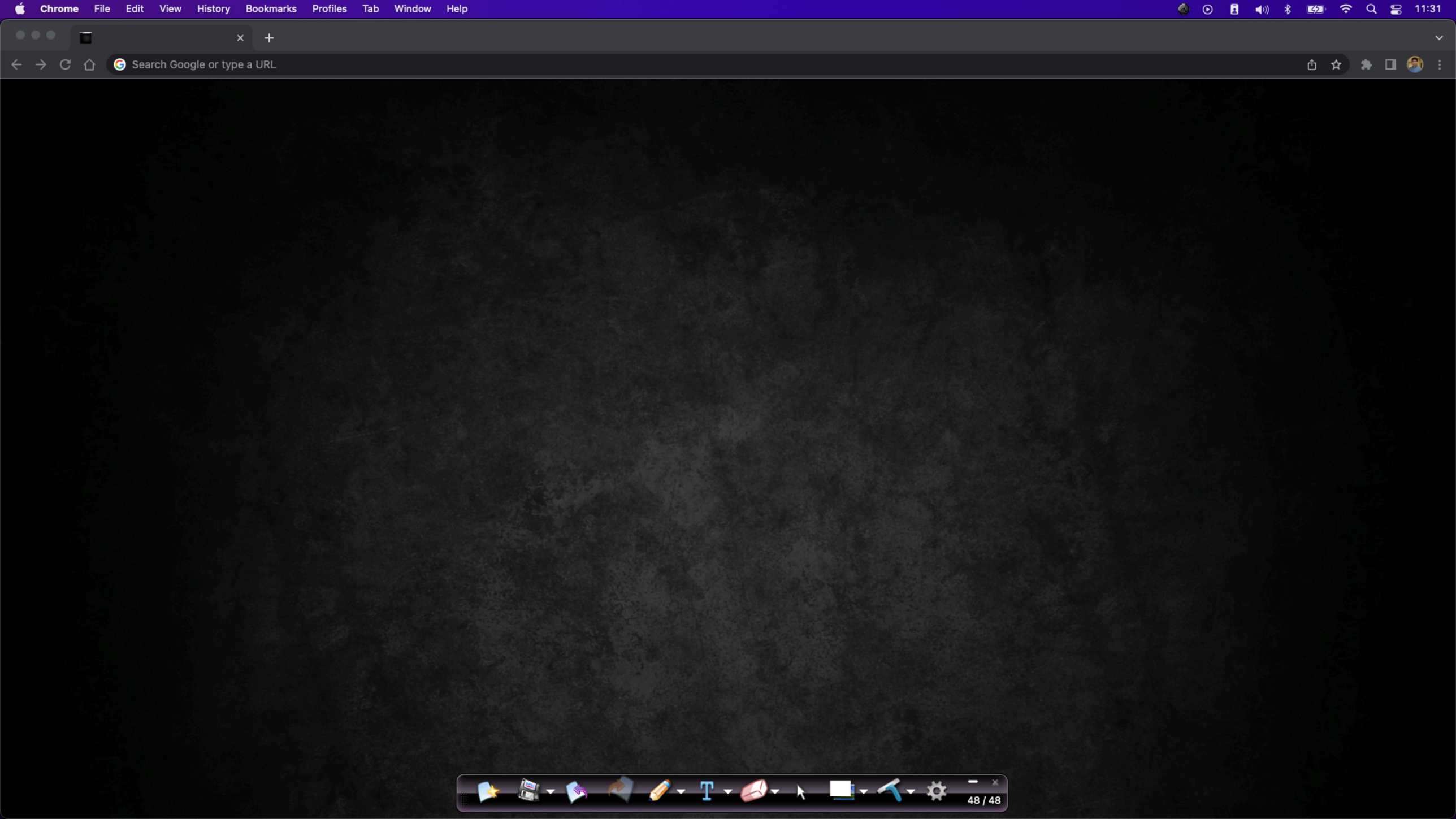


V.V. expensive

→ hardware (GPUs)

→ lot more data → lots of params ✓

2 hrs + 30 min
(Q&A)


Chrome File Edit View History Bookmarks Profiles Tab Window Help
11:31
Search Google or type a URL
48 / 48