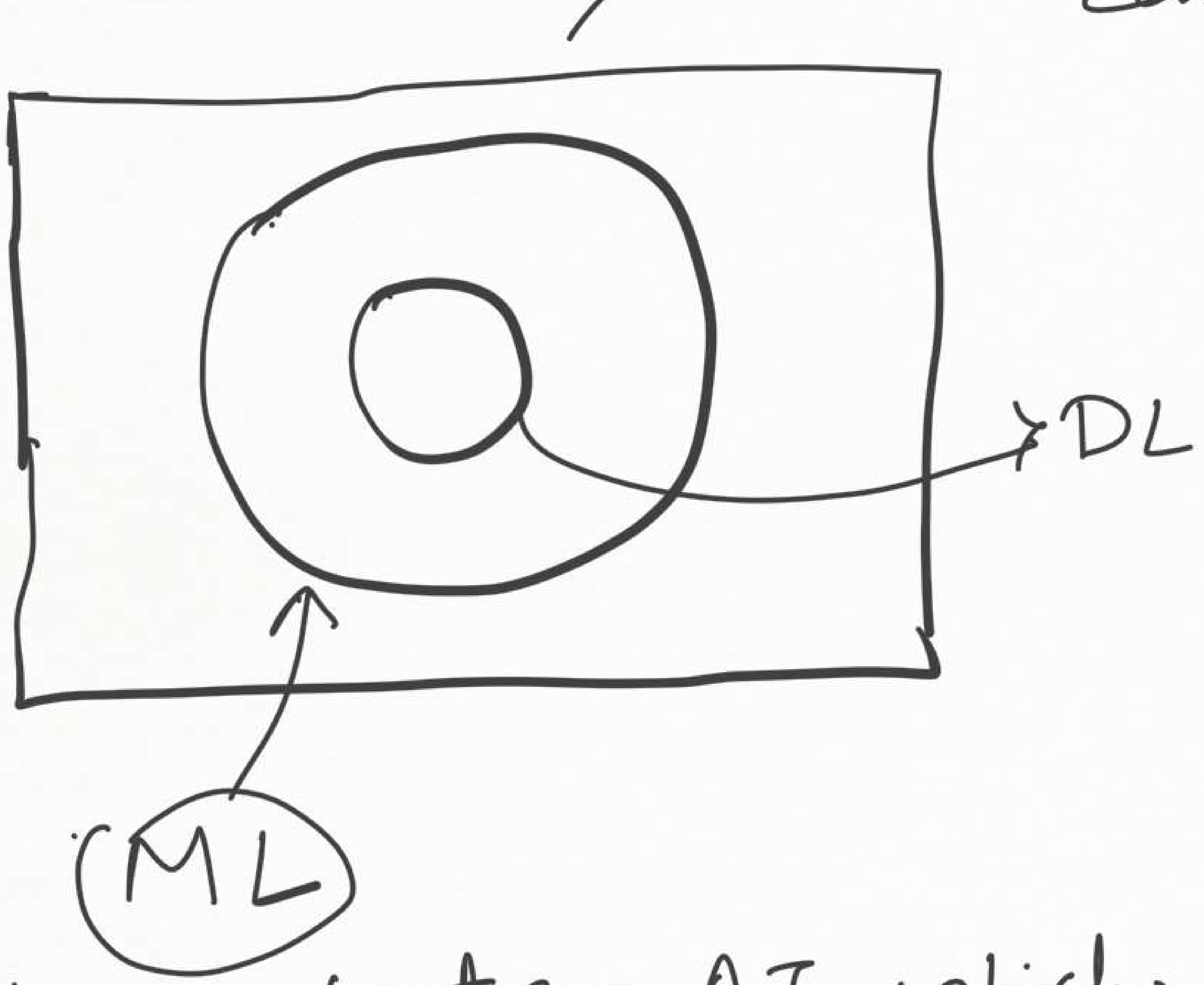
AI {Application which can do its own task without any human intervention}



→ Netflix App

→ Self Driving Cars

→ ANPR

→ Alexa, Sofia

→ ChatGPT

It is a subset on AI, which can make pattern on historical data often for making predictions.

CNN RNN ML

DL → "Neural Networks" (Geoffrey Hinton) {2006}

(Mimic the human brain)

Structured

- .csv
- .excel
- .html

Unstructured Data

- Videos
- Text
- Audio
- Images

Q.) Why Deep Learning is becoming so popular?

⇒ 2005 → ORKUT, facebook, Insta, Twitter

Data ↑↑↑ exponentially

2016 "PetaBytes"

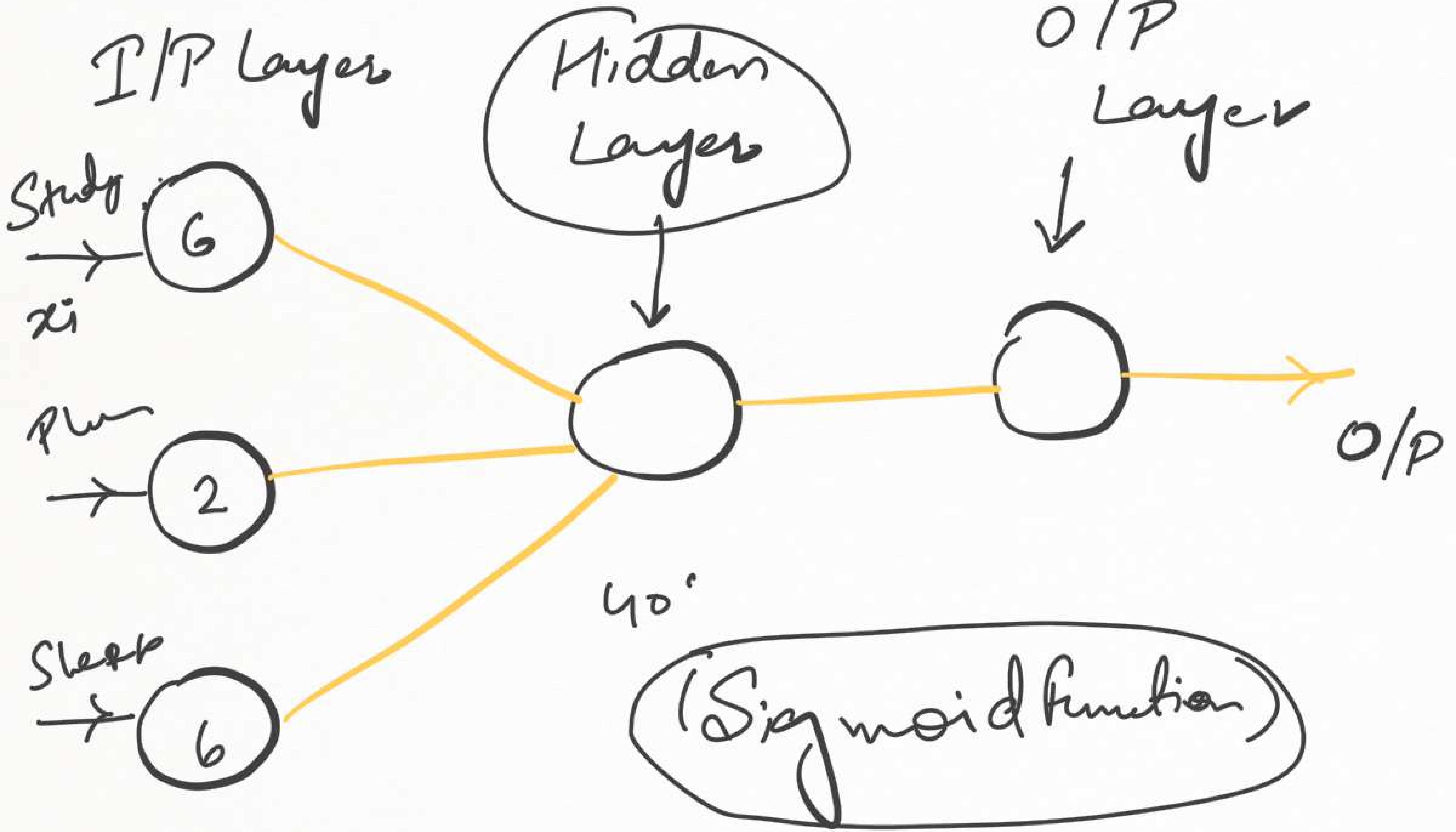
Spo

GPU's

TPU

Hardware Ac

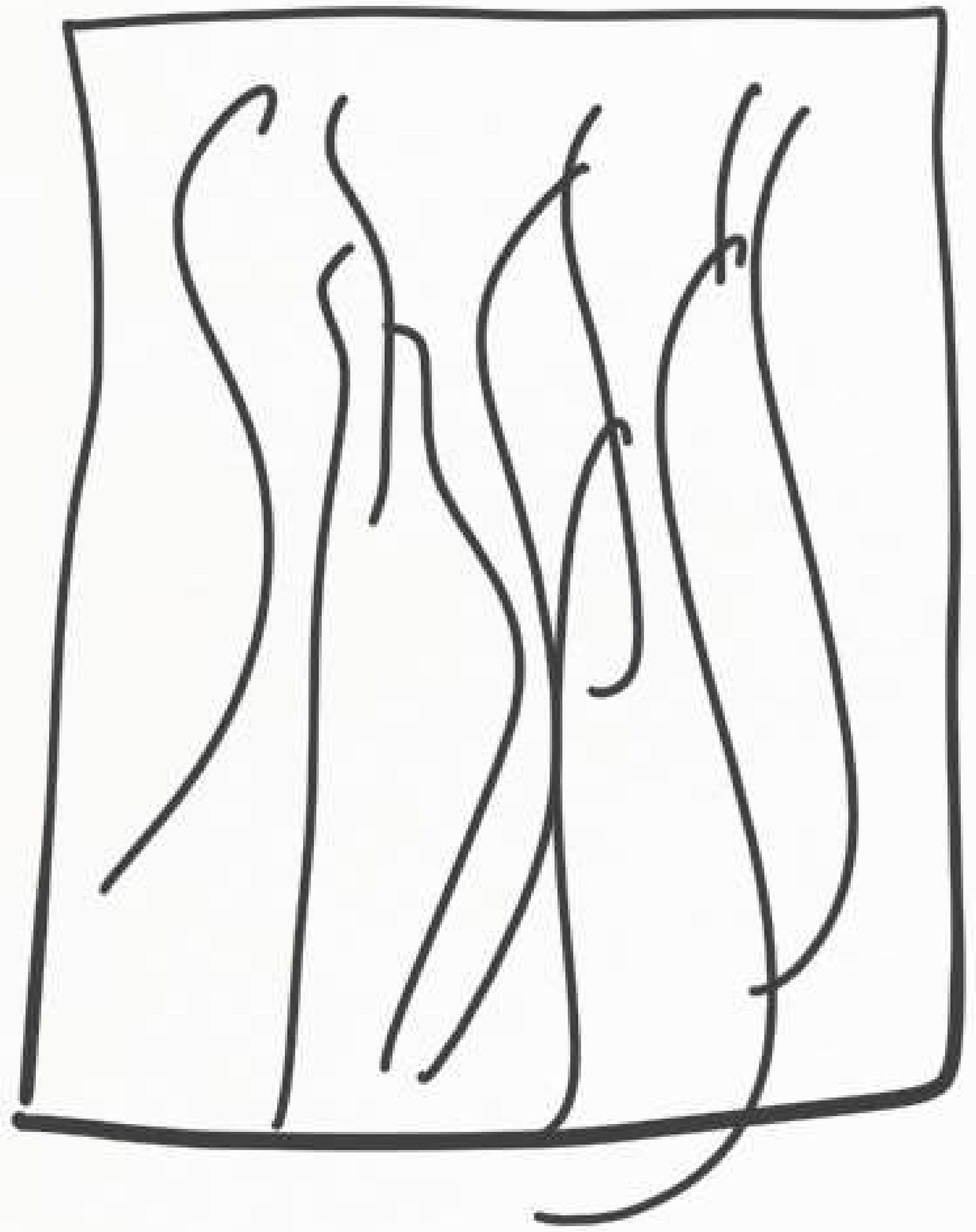
Perceptron {Single Layer Neural N/w}

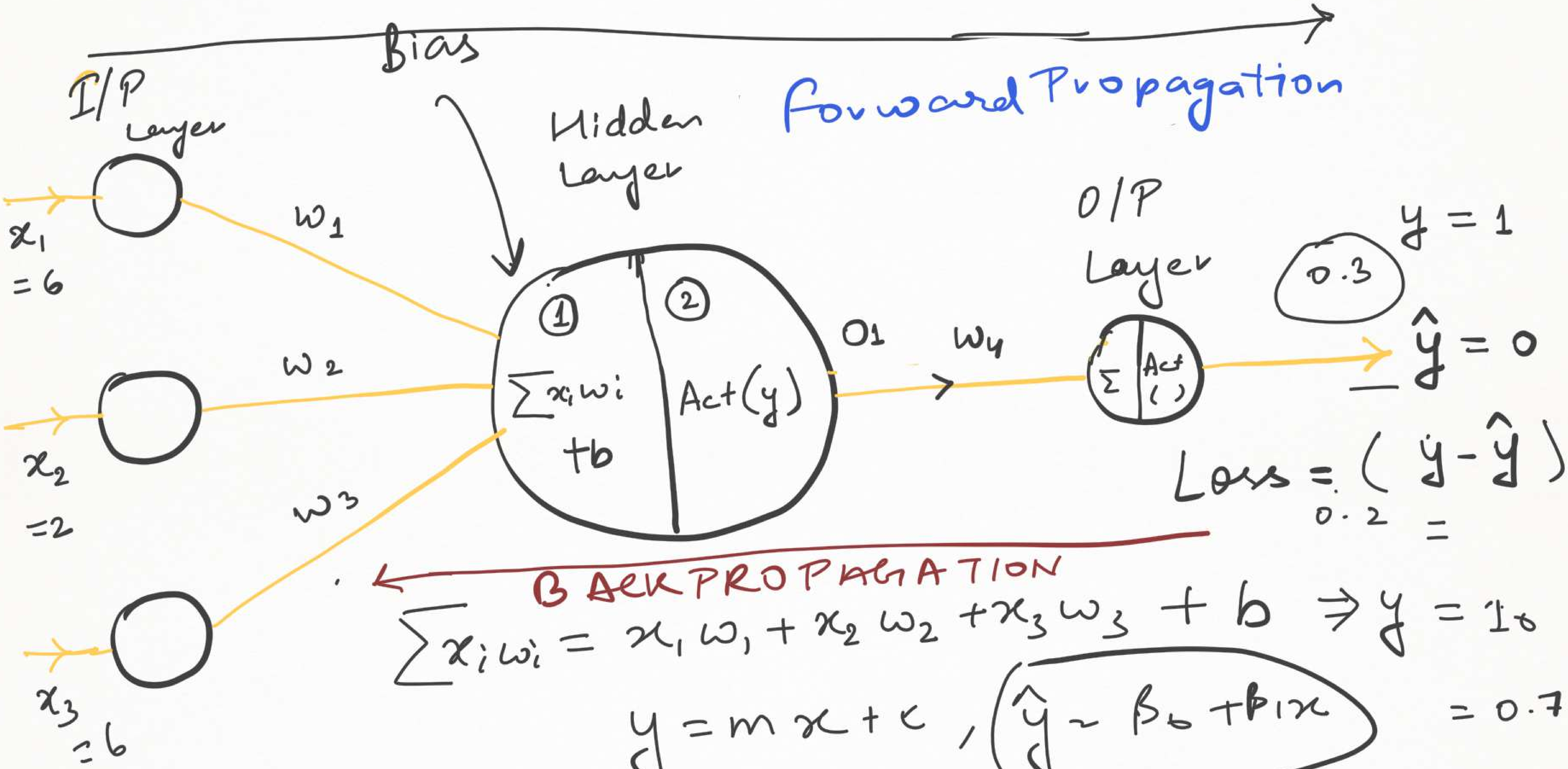


Example

Feature			Target
Study	Play	Sleep	P/F
6	2	6	1
2	5	8	0
5	3	7	1

1-year

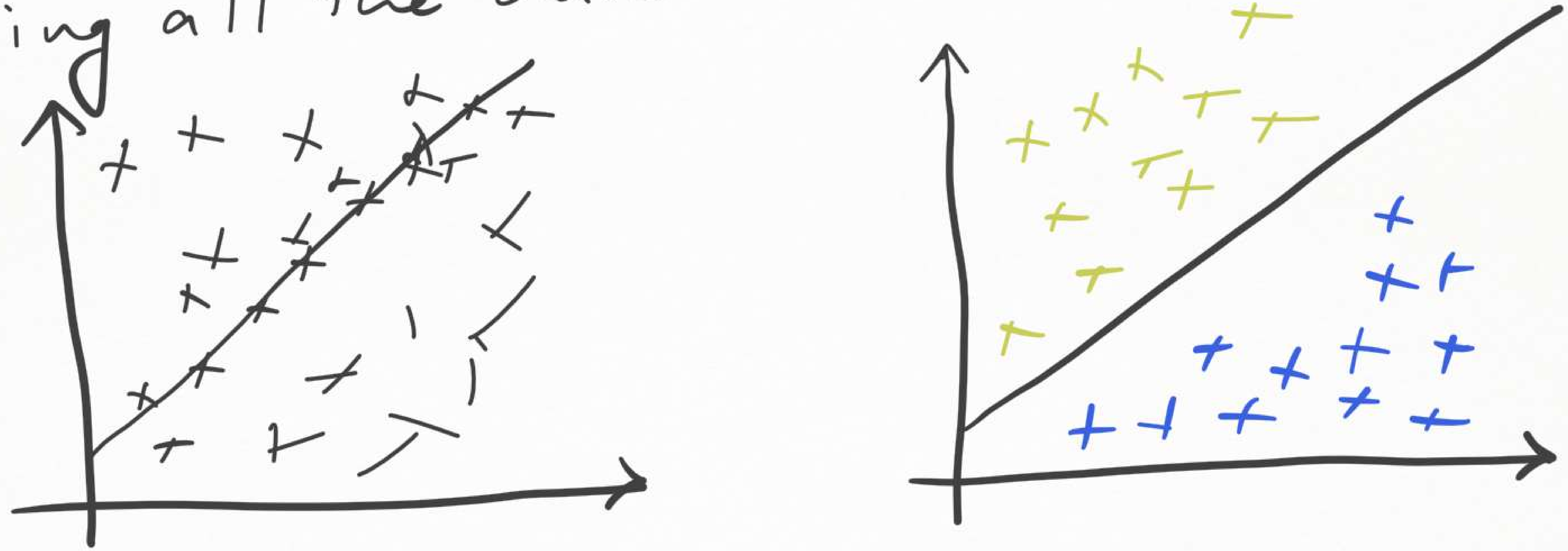




Bias
I/P Layer
Hidden Layer
Forward Propagation
$x_1 = 6$
$x_2 = 2$
$x_3 = 6$
w_1
w_2
w_3
①
②
$\sum x_i w_i + b$
Act(y)
O_1
w_4
O/P Layer
Σ
Act ( )
$y = 1$
0.3
$\hat{y} = 0$
$Loss = (y - \hat{y})$
0.2
=
BACKPROPAGATION
$\sum x_i w_i = x_1 w_1 + x_2 w_2 + x_3 w_3 + b \Rightarrow y = 1_0$
$y = mx + c$, $\hat{y} = \beta_0 + \beta_1 x$
$= 0.7$

Activation functions

→ Support non-linear properties.
→ Bring all the data into same scale.



Sigmoid A f

$$Z = \frac{1}{1 + e^{-y}} = \frac{1}{1 + e^{-(\Sigma x_i w_i + b)}}$$

$$\downarrow$$

$$= 0 \text{ to } 1$$

$\geq 0.5 \Rightarrow 1$

$< 0.5 \Rightarrow 0$

Forward Propagation

1.) I/P Layer

2.) weights

3.) Activation functions

4.) O/P Layer

5.) Loss Calculation

Back Propagation

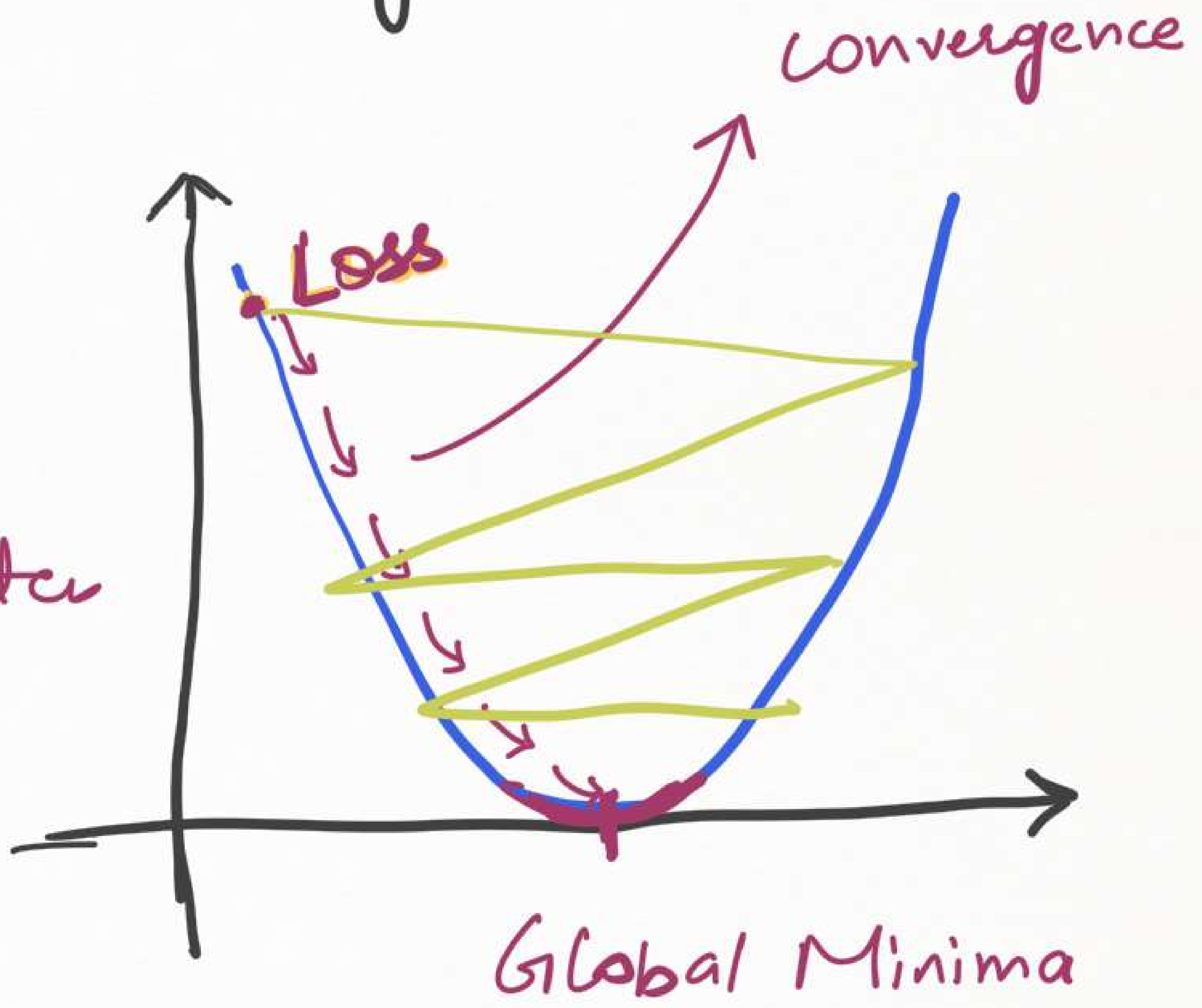
1. Update the weights

2. Optimizers

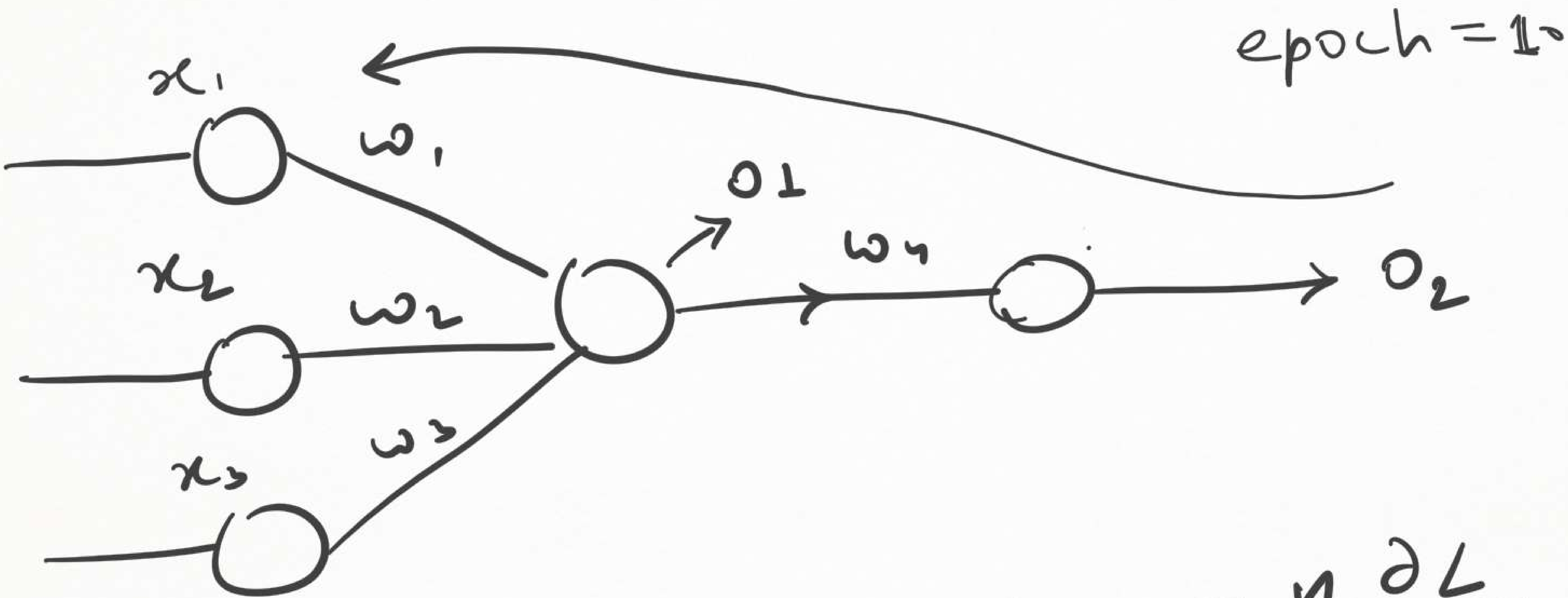
Weight Updation formula

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

$\eta \rightarrow$ Learning Rate $= 10^{-1}$ to 10^{-3}



* Learning Rate is a hyperparameter which controls the speed of convergence.

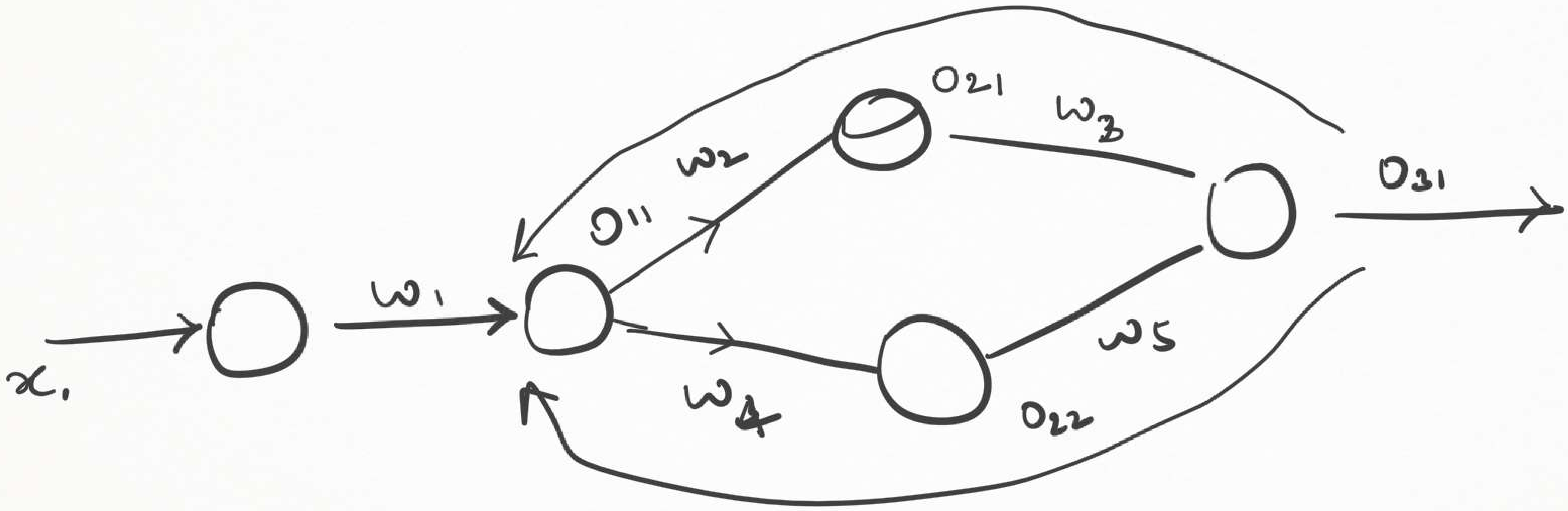


$$w_{4new} = w_{4old} - \eta \frac{\partial L}{\partial w_{4old}}$$

$$\frac{\partial L}{\partial w_{4old}} = \frac{\partial L}{\partial O_2} \times \frac{\partial O_2}{\partial w_{4old}} \Bigg\} \text{chain Rule differeniation}$$

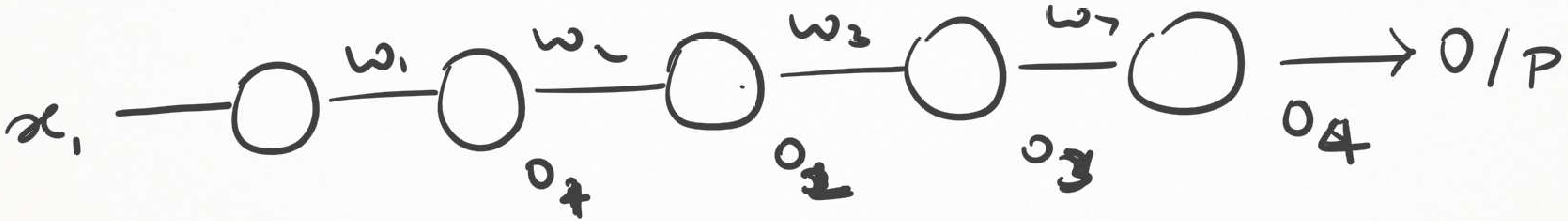
$$\boxed{\frac{\partial L}{\partial w_{1new}} = \frac{\partial L}{\partial O_2} \times \frac{\partial O_2}{\partial O_1} \times \frac{\partial O_1}{\partial w_{1old}}}$$

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



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

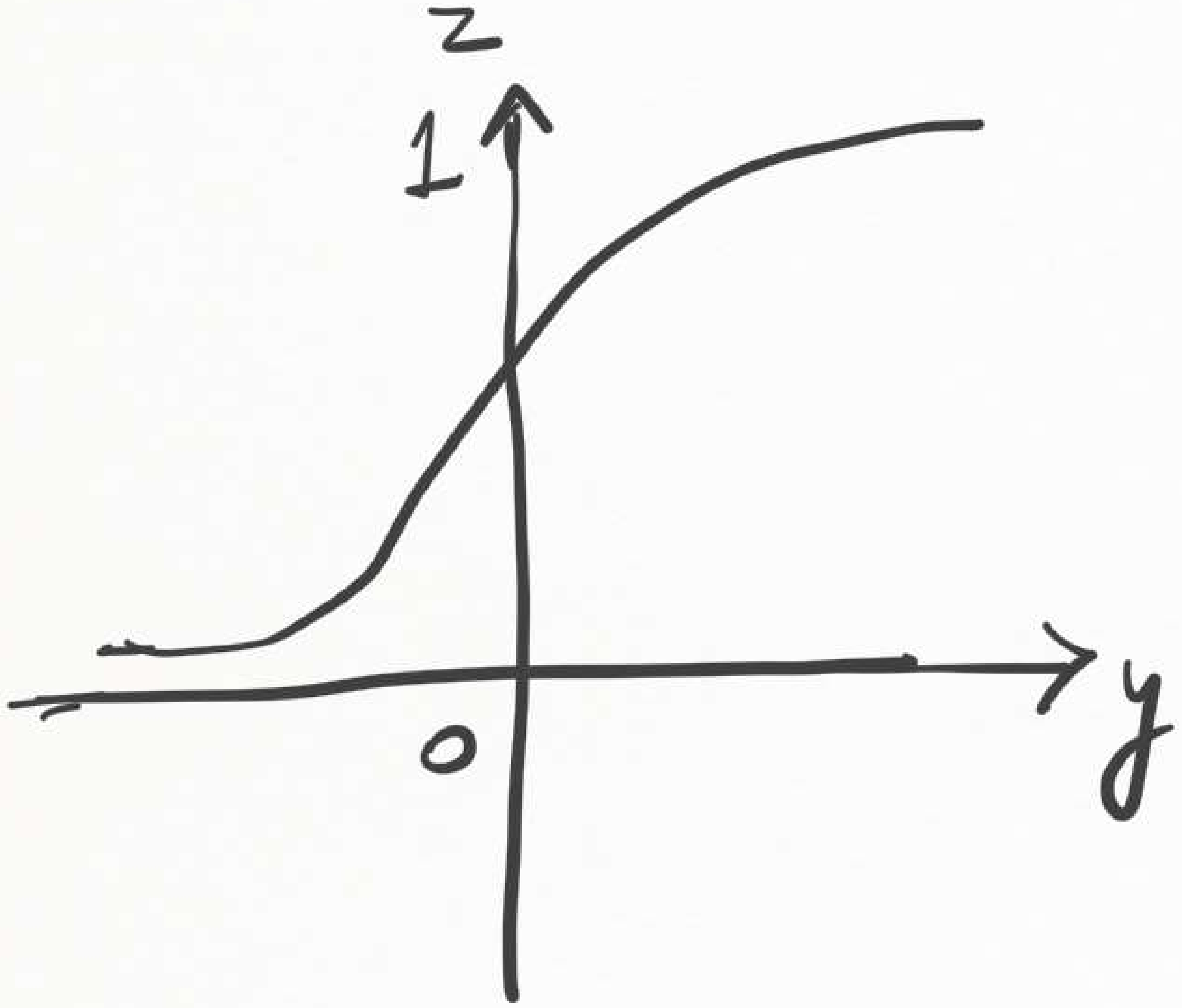
$$\frac{\partial L}{\partial w_{1old}} = \frac{\partial L}{\partial O_{31}} \times \frac{\partial O_{31}}{\partial O_{21}} \times \frac{\partial O_{21}}{\partial O_{11}} \times \frac{\partial O_{11}}{\partial w_{1old}}$$



$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial O_4} \times \frac{\partial O_4}{\partial O_3} \times \frac{\partial O_3}{\partial O_2} \times \frac{\partial O_2}{\partial O_1} \times \frac{\partial O_1}{\partial w_1}$$

$$= 0.25 \times 0.25 \times 0.25 \times 0.25 \times 0.25$$

$$= 0.0000625 \simeq 0$$



$$\left(\text{small value}\right) \times \left(\text{very small value}\right)$$

0.01×0.0000625

$=$

very very very small

$z = 0 \text{ to } 1$

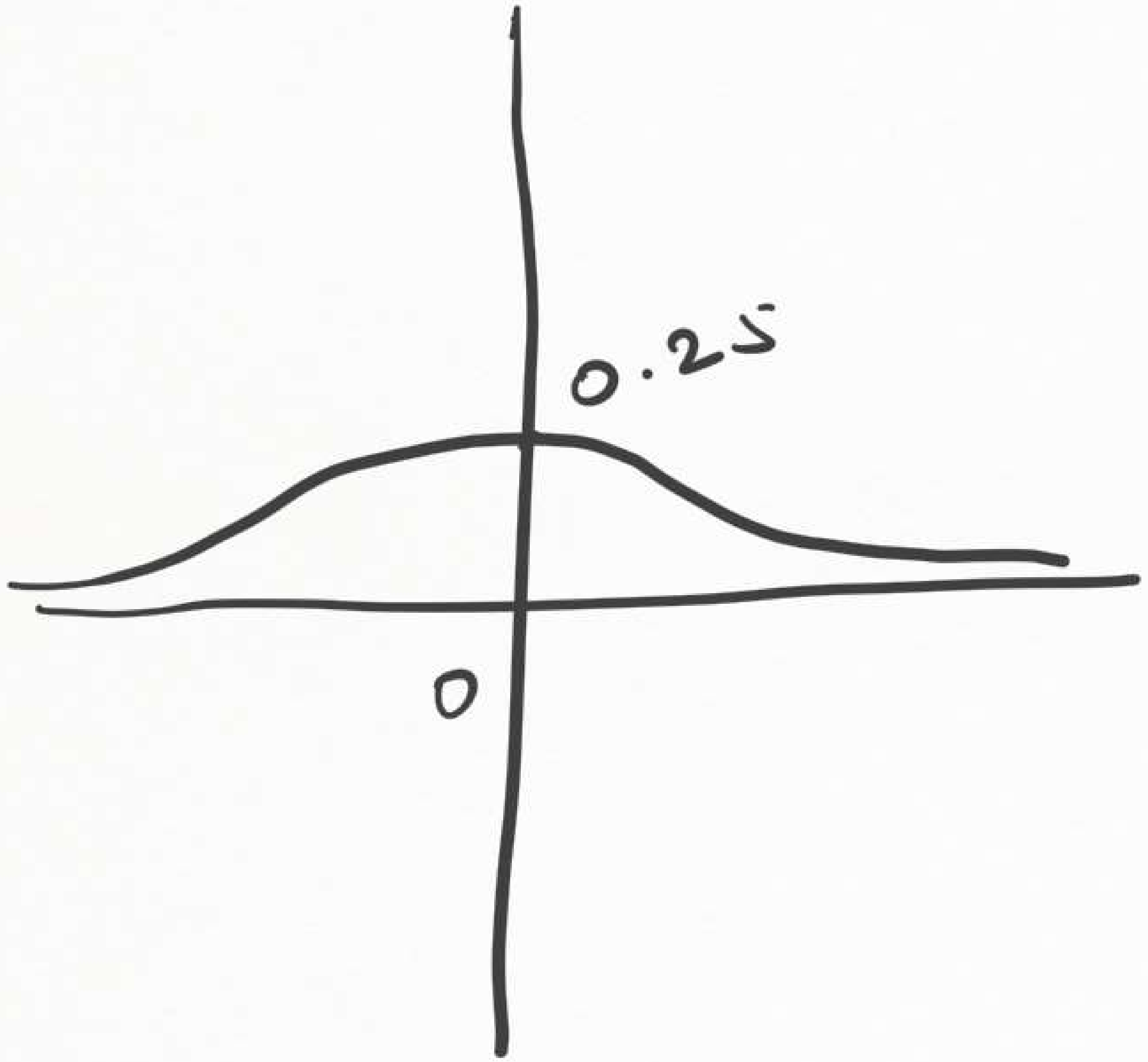
$$z = \frac{1}{1+e^{-x}}$$

Derivative $(z) = 0$ to 0.25

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

$$\boxed{w_{new} \approx w_{old}}$$

{"Vanishing Gradient Problem"}



0.25

"Sigmoid Af"

FP

$\Sigma \to Af \to O/P \to Loss$

Update Weight $\Longrightarrow$ Optimizers

BACKPROPAGATION

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

z

$$\sum w_i x_i = 17$$

$$\sigma(\cdot) = \frac{1}{1+e^{-17}} = 0.7$$

$$\sum w_i x_i = -17 \qquad \max(0,-17) = 0$$

$$\mathrm{ReLU}(x) = \max(0,17)$$

$$= 17$$

Softmax AF

1, 2, 3, 4, 5

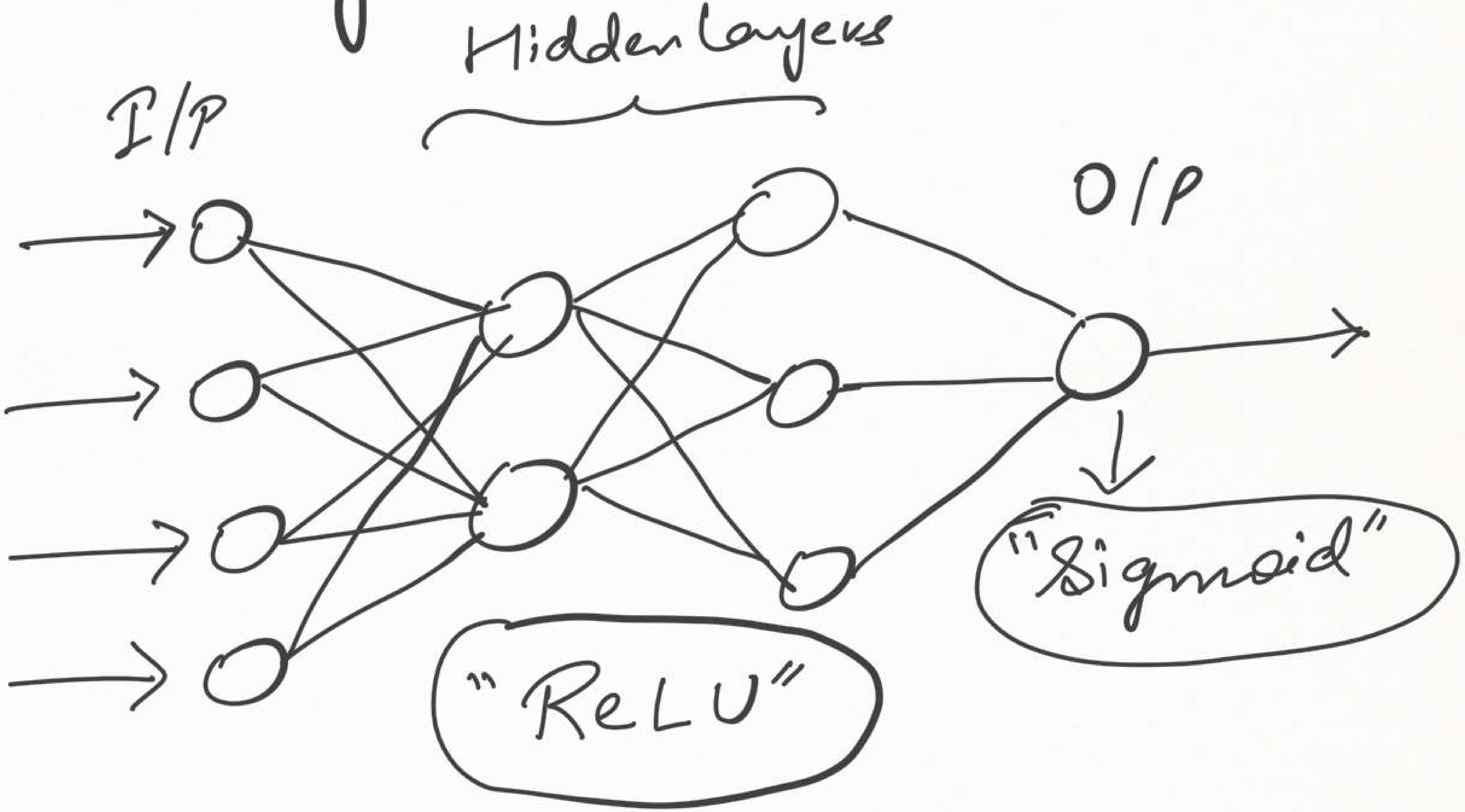
$$S(i) = \frac{e^{i}}{\sum e^{i}}$$

$$S(i=2) = \frac{e^{2}}{e^{1} + e^{2} + \dots e^{5}}$$

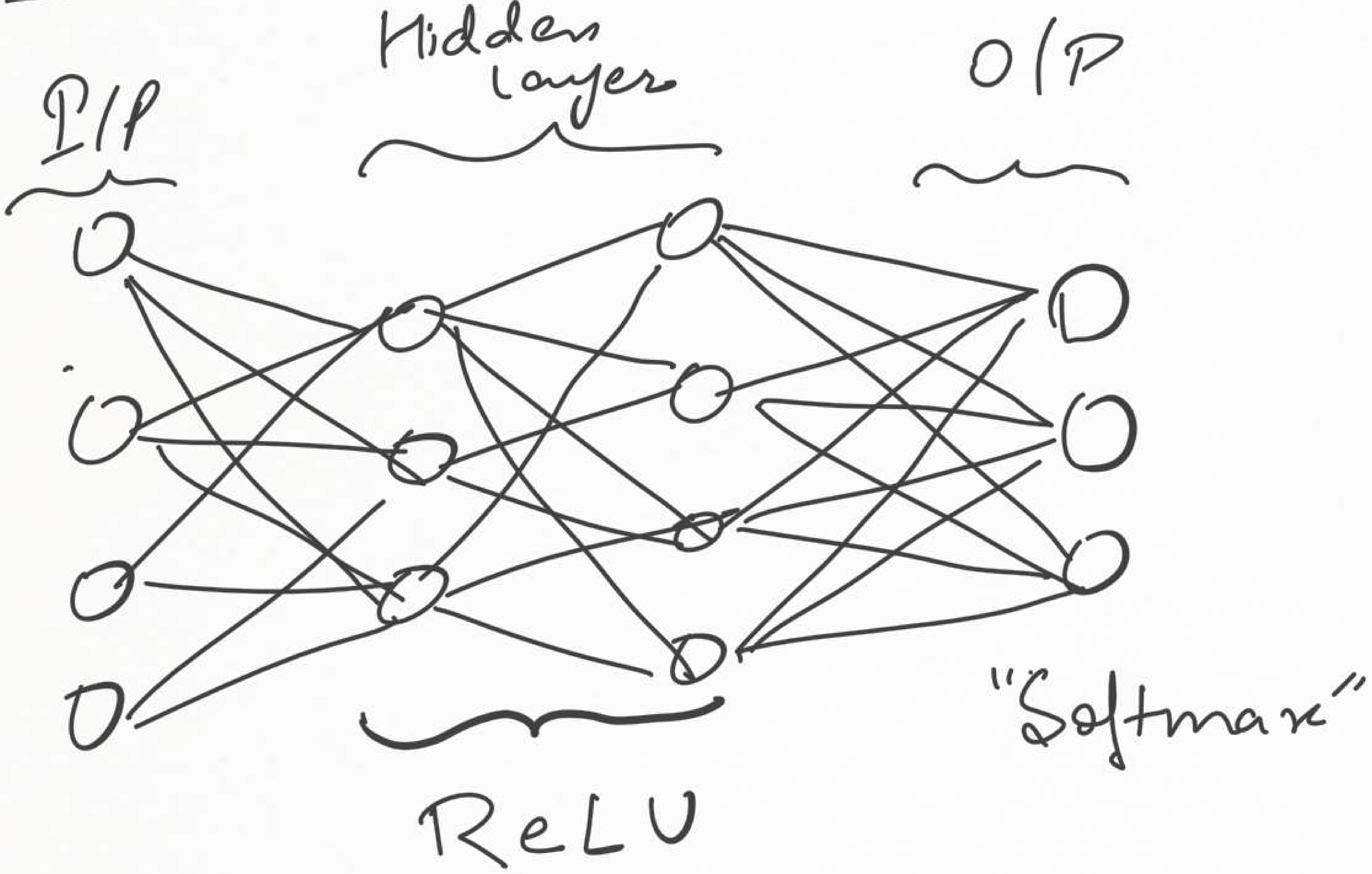
$$S(i=1) = \frac{e^{1}}{e^{1} + e^{2} + e^{3} + e^{4} + e^{5}} =$$

1.) Sigmoid
2.) Tanh
3.) ReLU
4.) Leaky ReLU
5.) ELU
6.) PReLU
7.) Softmax

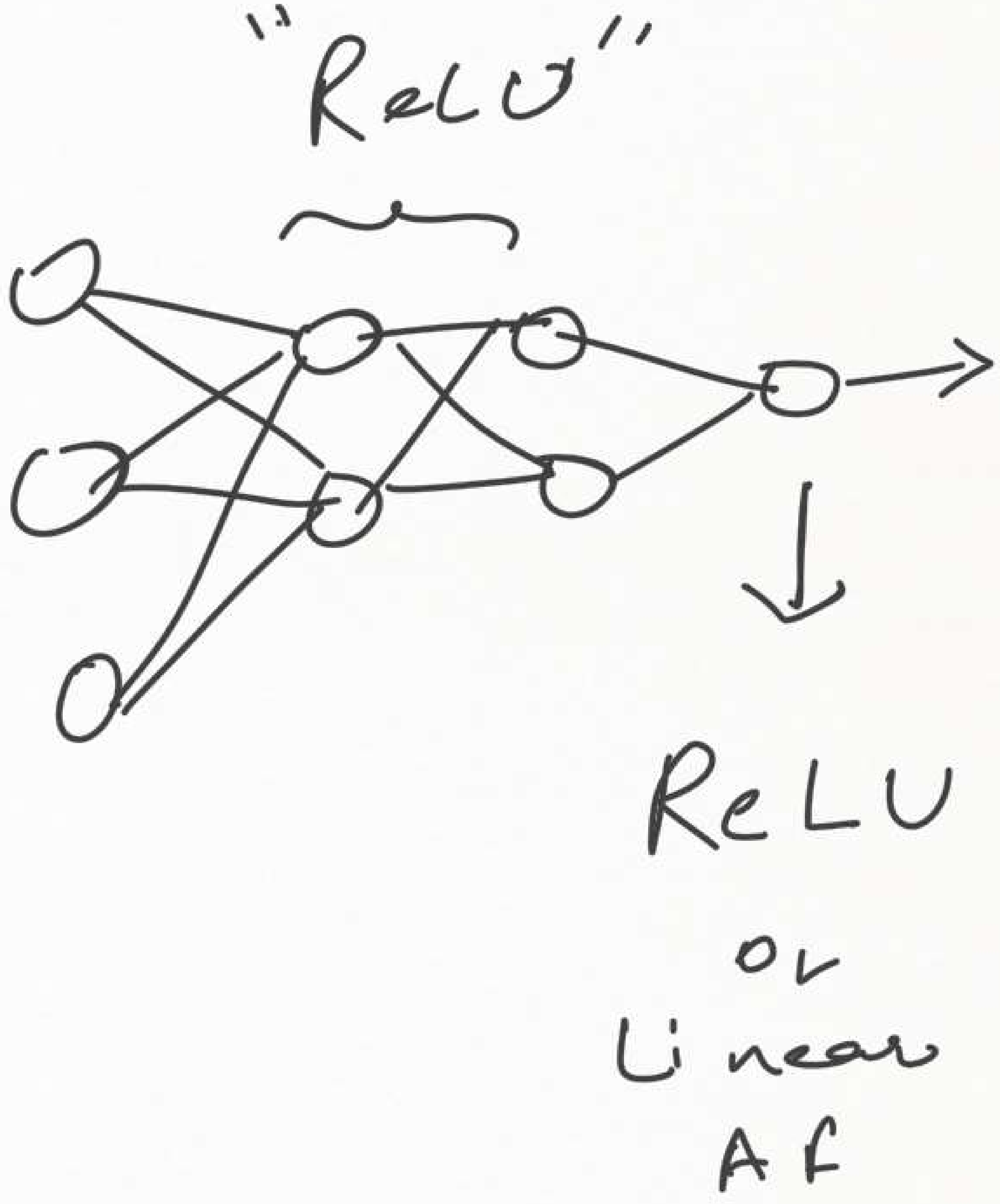
Binary Classification



MultiClass Classification



Regression



Loss or cost

$\downarrow$

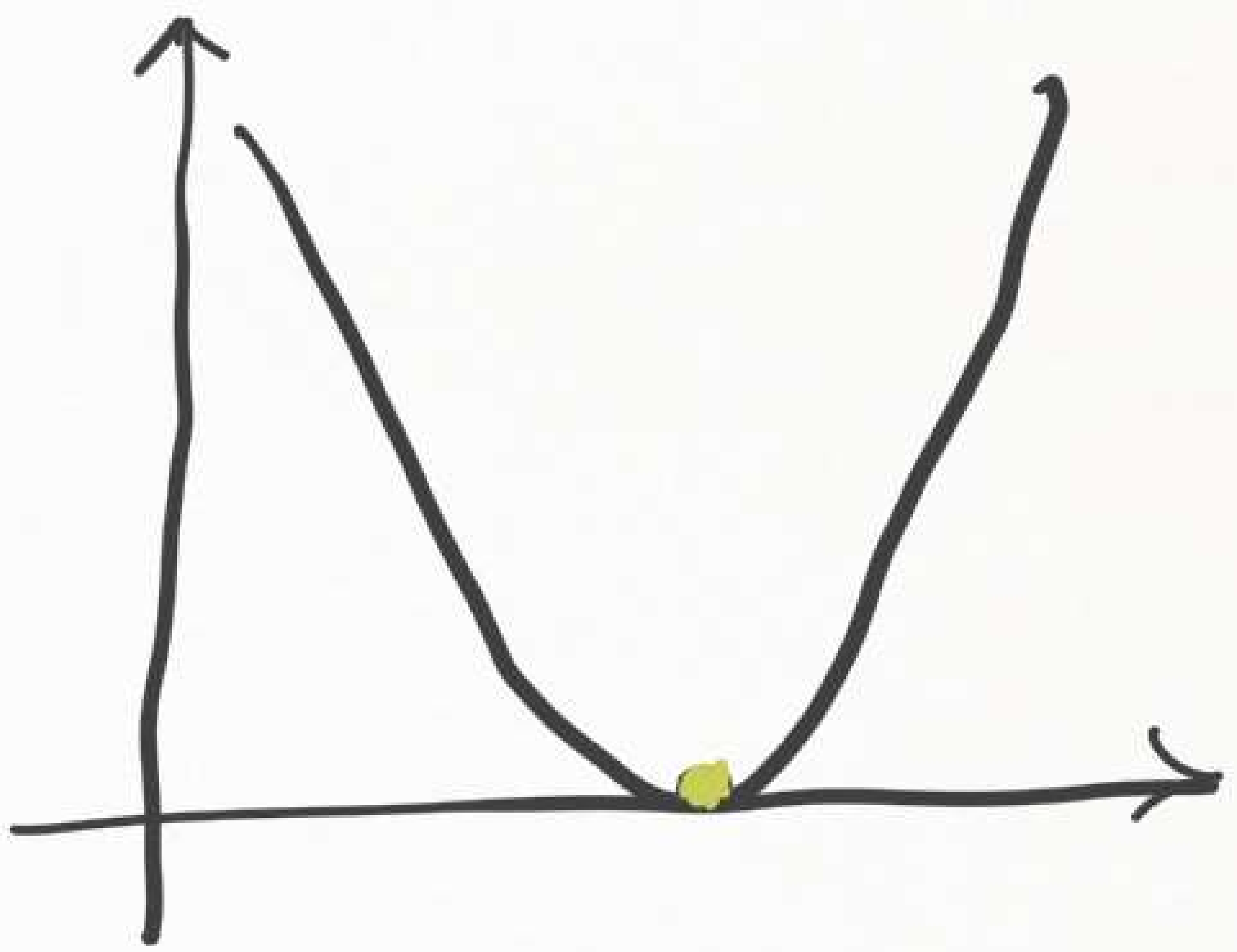
MSE

$$\text{Error} \begin{cases} \text{Loss} = \frac{1}{2}(y-\hat{y})^2 \\ \text{cost} = \frac{1}{2n}\sum_{i=1}^{n}(y-\hat{y})^2 \end{cases}$$

Regression Task

1.) Mean Squared Error (MSE)

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



Adv.

1.) Differentiable
2.) Convergence is fast
3.) It has only 1 local or global minima.

Disadvantage

1.) Not robust to an outlier.

Mean Absolute Error

$$\frac{1}{2n}\sum_{i=1}^{n}|y-\hat{y}|$$

1 Robust to an outlier.

③ Huber Loss

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

10 → hyper parameter

Classification

CROSSENTROPY

- → (Two classes) Binary cross entropy (sigmoid cross ")
- → Categorical cross entropy (Softmax crossent.) (Multiple classes)

Binary Cross Entropy -

$$\boxed{\text{Loss} = -y * \log(\hat{y}) - (1-y) * \log(1-\hat{y})}$$

$$\text{Loss} = \begin{cases} -\log(1-\hat{y}) & \text{if } y=0 \\ -\log(\hat{y}) & \text{if } y=1 \end{cases}$$

$$\hat{y} = \frac{1}{1+e^{-z}}$$

$$\text{cost}(J) = -\left[\sum_{i=1}^{n} y * \log(\hat{y}) + (1-y) * \log(1-\hat{y})\right]$$

$$= \cancel{7}.3 \approx 0$$

$$\hat{y} = \frac{1}{1+e^{-z}}$$

Categorical Cross Entropy (Multi-class Classification)

f_1	f_2	f_3	O/P	fever	Malaria	J
2	6	4	fever	1	0	0
5	1	8	Malaria	0	1	0
7	3	9	Jaundice	0	0	1

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

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

$$\sigma(z) = \frac{e^{z_i}}{\sum_{i=1}^{c} e^{z_i}}$$

$(\hat{y})$

$\sigma(f) = 0.2$

$\sigma(m) = 0.1$

$\sigma(J) = 0.7$

Conclusion

Regression
- ReLU (HL)
- ReLU or linear (OL)
- MSE, MAE or Huber

Classification

- **Binary**
  - ReLU (HL)
  - Sigmoid (OL)
  - Binary cross entropy
- **Multiclass**
  - ReLU (HL)
  - Softmax (OL)
  - Categorical crossentropy

Optimizers

1. Gradient Descent (Batch)
2.) Stochastic GD (SGD)
3.) Mini-Batch GD (MBGD)
4.) SGD with momentum
5.) Adagrad
6.) RMSProp
7.) Adam

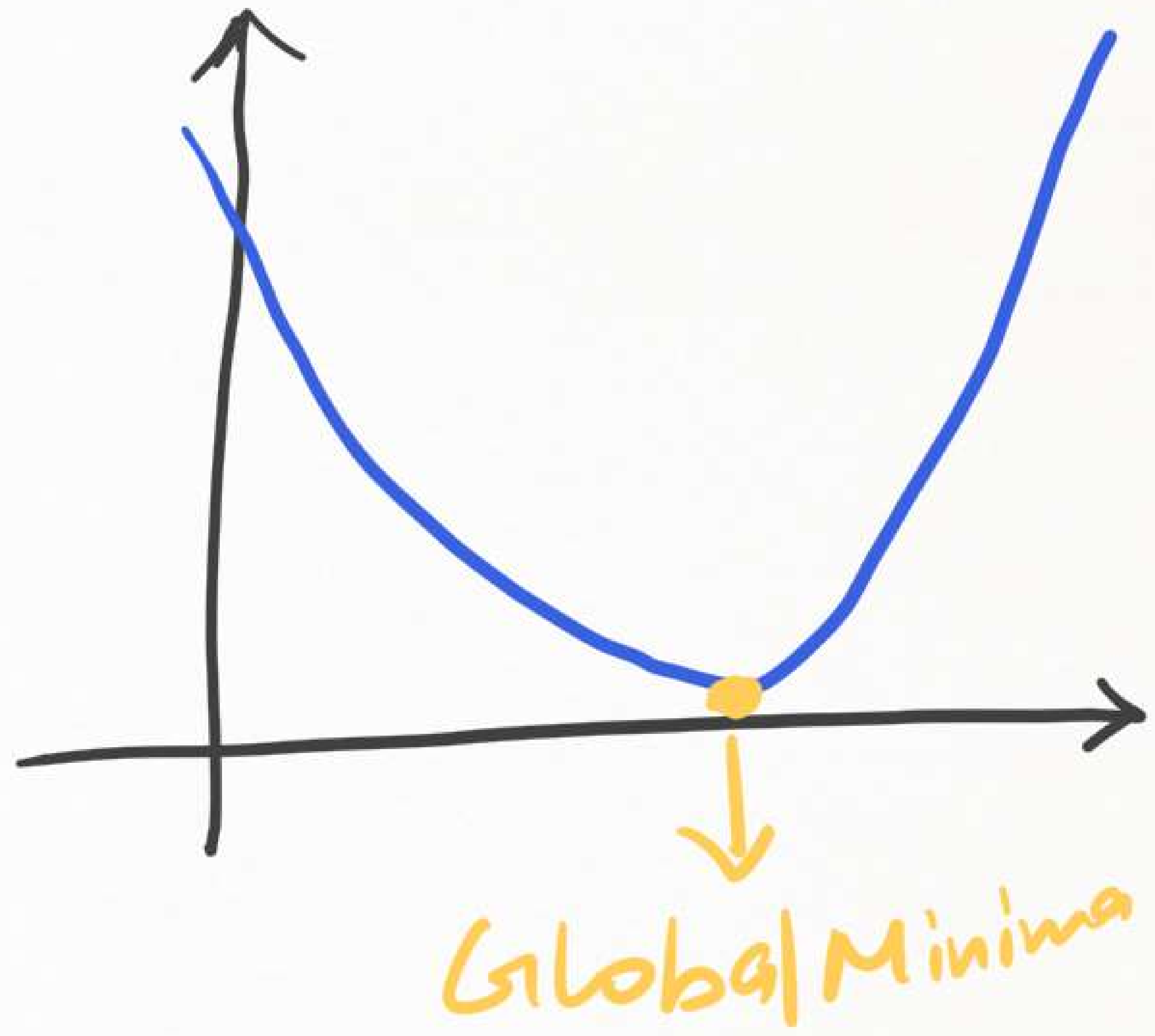
① Gradient Descent (Batch)

Weight Updation formula

$\eta \rightarrow$ Learning Rate

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

$\eta = 0.1$ to 0.001



Batch vs. Epoch vs. Iterations

$n = 1000$

Forward Propagation

1 Epoch

Back Propagation

1M

Dist.

1 Resource Extensive {Huge RAM}

1000 rows

100

Iter 1
(10)

$$\frac{10\cancel{0}\cancel{0}}{1\cancel{0}\cancel{0}} = 10$$

100

Iter 1
(10)

2
(10)

⋮

10
(10)

② Stochastic Gradient Descent (SGD) 1 M

Epochs

→ RAM ↓↓

1 record → ← } Iteration 1

Update weight

Dist.

1 record → ← } It 2

→ Time consuming.

→ convergence will be very slow.

③ Mini Batch SGD

1 M rows

Batch_size = 1000

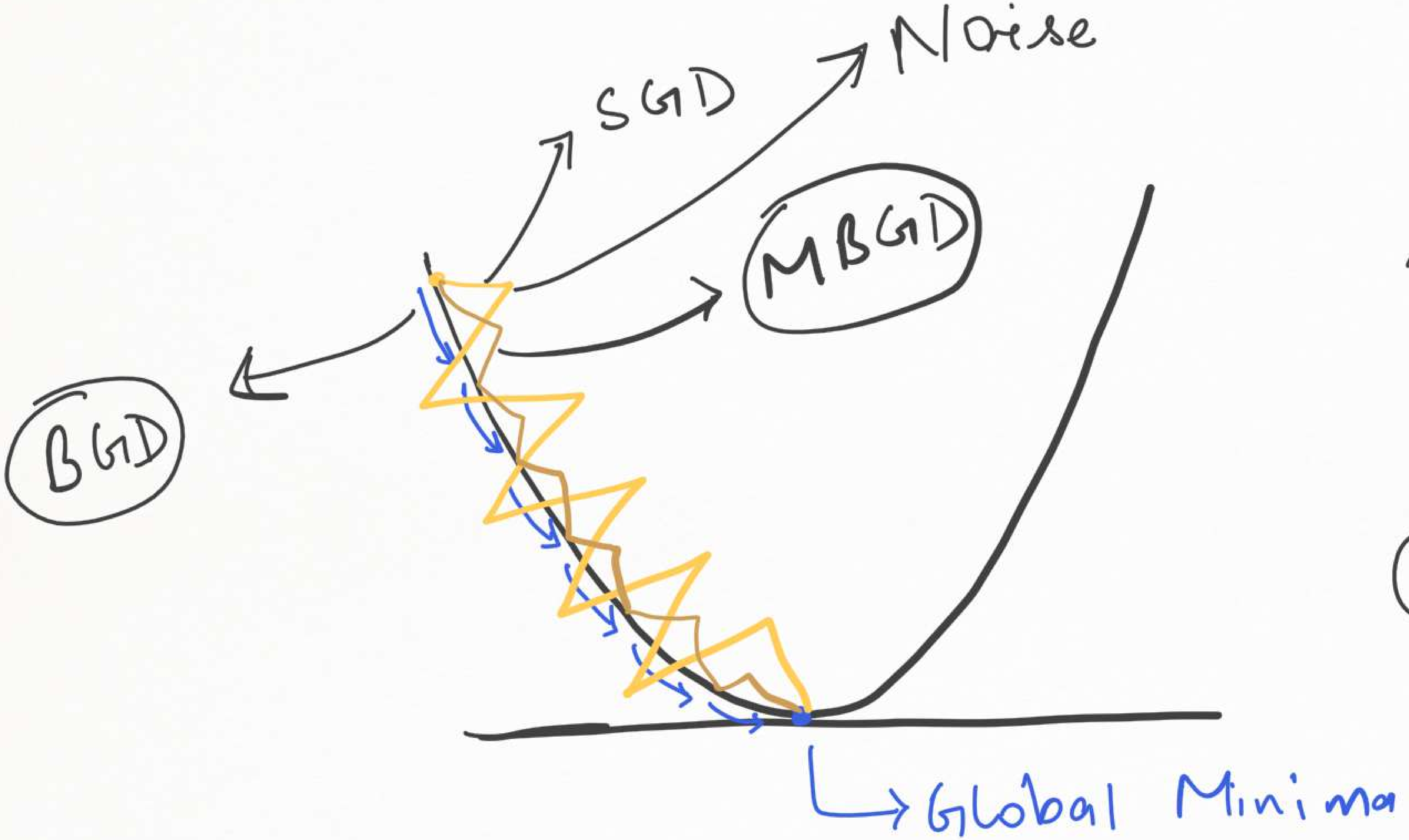
No. of iteration = $\frac{1000000}{1000}$

Epoch = 1

→ Resource Intensive

→ Convergence will be better

→ Time complexity will improve.



Reduce Noise

$\Downarrow$

"Momentum"

④ SGD with Momentum

w_1, w_2, w_3

$w_4 = w_3 - \eta \frac{\partial L}{\partial w_3}$

w_5

"Exponential Weighted Average"

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

t = current time

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

$t-1$ = previous timestamp

Exponential Weight Average

$t_1 \quad t_2 \quad t_3 \quad t_4 \quad - - - \cdot — t_n$

$a_1 \quad a_2 \quad a_3 \quad a_7 \quad - - \quad \cdot - a_n$

$\beta = 0 \text{ to } 1$

$\beta = 0.95$

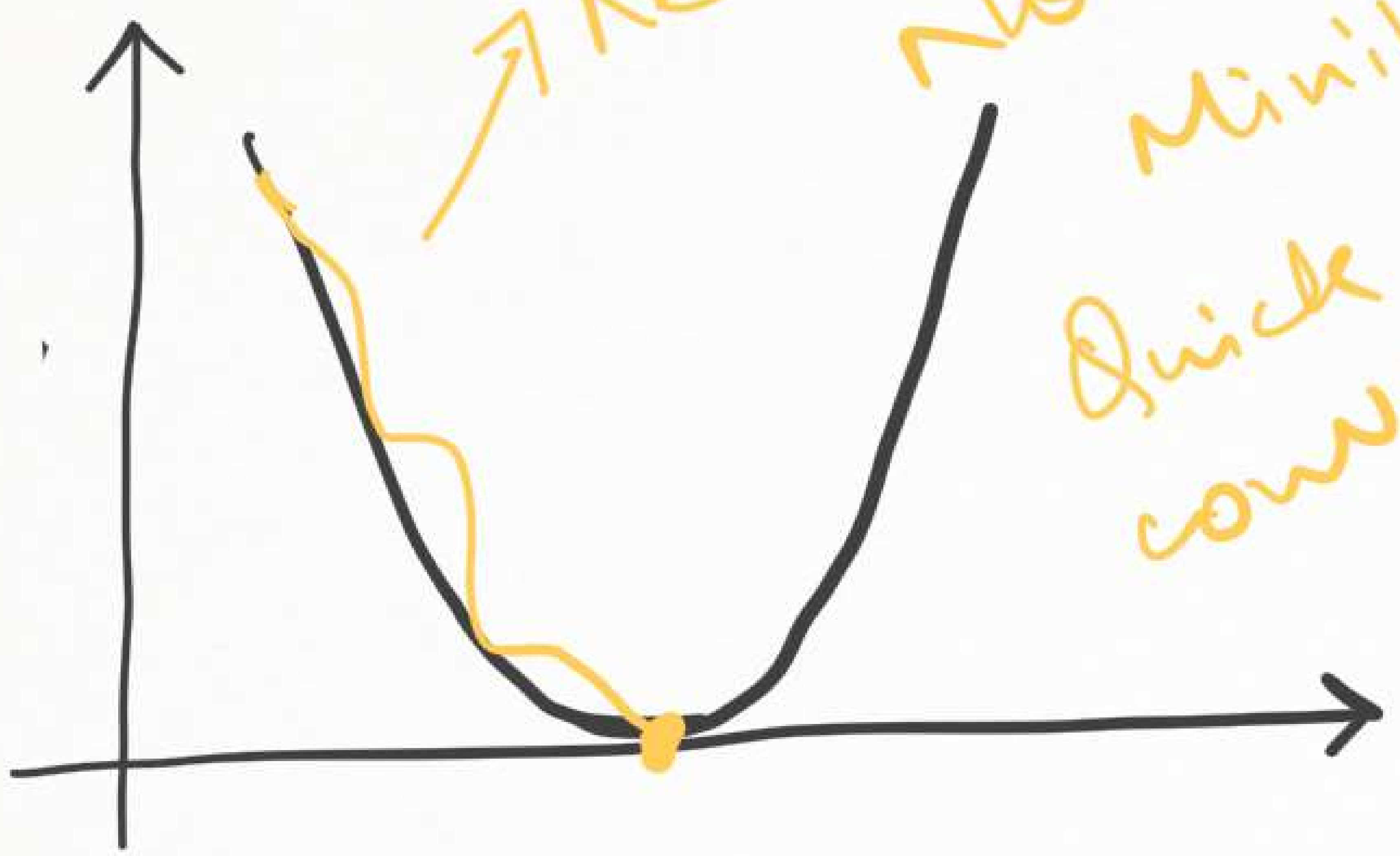
$V_{t_1} = a_1$

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

$= 0.95 \times a_1 + 0.05 \times a_2$

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

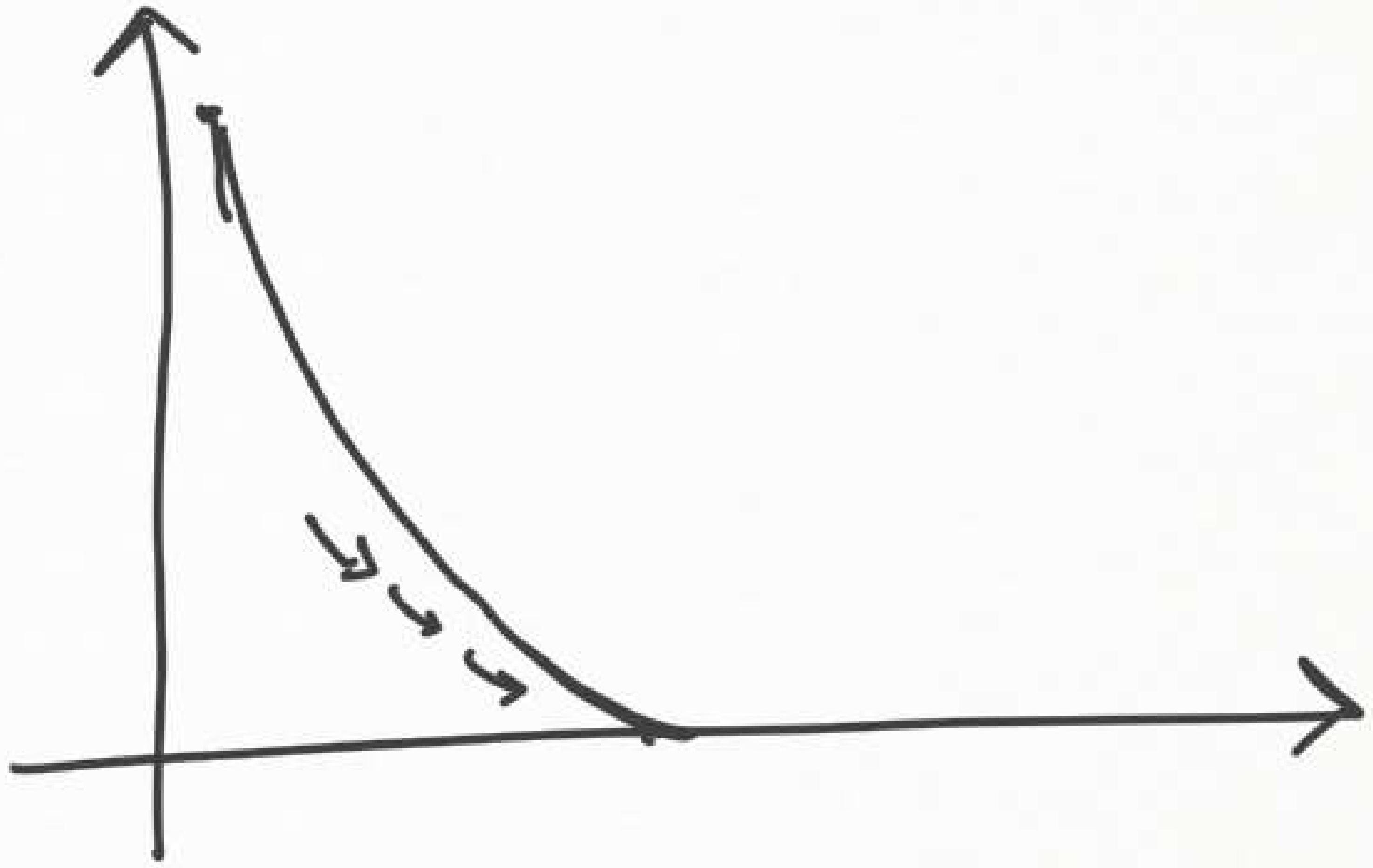
"SGD with Momentum"



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

$\eta \rightarrow$ fixed value

$0.1 = 0.1$



5 Adagrad $\rightarrow$ Adaptive Gradient Descent

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

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

$\epsilon \rightarrow$ small value

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

⑥ Adadelta & RMSProp

$\beta = 0.95$

$S_{dw} = \beta * S_{dw_{t-1}} + (1-\beta)\left(\frac{\partial L}{\partial w_{t-1}}\right)^2$

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

$0.95 * S_{dw_{t-1}} + 0.05 \times$

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

(7) Adam Optimizer (Best Optimizer)

Adaptive + Momentum

(RMSProp + SGD with Mom.)

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

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

$$H_{11} = \sum x_1 \times w_{11} + x_2 \times w_{21} + x_3 \times w_{31} + \beta_1 = 4$$

$\max(0, x) = 4$

$$H_{12} = \sum x_1 w_{12} + x_2 * w_{22} + x_3 * w_{32} + \beta_2 = 3$$

$\max(0, x) = 3$

"Nested Loop"

Epoch

Batch Size