
5 DAYS
DEEP LEARNING
LIVE COMMUNITY
SESSION

Day 1 : Deep Learning

7pm - 9pm IST

Motive : { Clear their Basics, Maths, Interview preparation }

Agenda

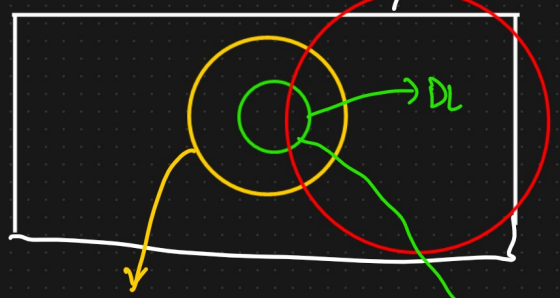
1. Deep Learning → Perceptron { AI Vs ML vs DL vs DS }
2. Forward Propogation
3. Backward Propogation
4. Loss function
5. Activation functions
6. Optimizers.

Prerequisite

1. Python
2. ML
3. Stats

AI VS ML VS DL VS DS

Deep Learning → DL



{ ML is a subset of AI }

Goal { application which can do its own task without any human Intervention }

1. Netflix App
2. Self Driving Cars
3. Amazon Application
4. Sofia

RTX 3090

ML → Statstool to analyze the data, visualize the data, predictions, forecasting, clustering

Researchers (1958)

Multi Layered Neural N/w { Mimic the human brain }

{ Perceptron }

(*) Why Deep Learning Is Becoming Popular?

① 2005 → ORKUT, FACEBOOK, Instagram, WhatsApp, LinkedIn Twitter

DATA → Exponentially ↑↑↑

2008 → {BIG DATA} → Efficiently

Year {2013-2014}

Netflix

2013 → Company had huge amount of Data

⇓ {AI → popular}

Scanlon Products

Panasonic : AC's, TV's, Refrigerator {Data}

↓ ⇓

Model → Reduce the Electricity Bills ⇓

⇓

Subscription Basis

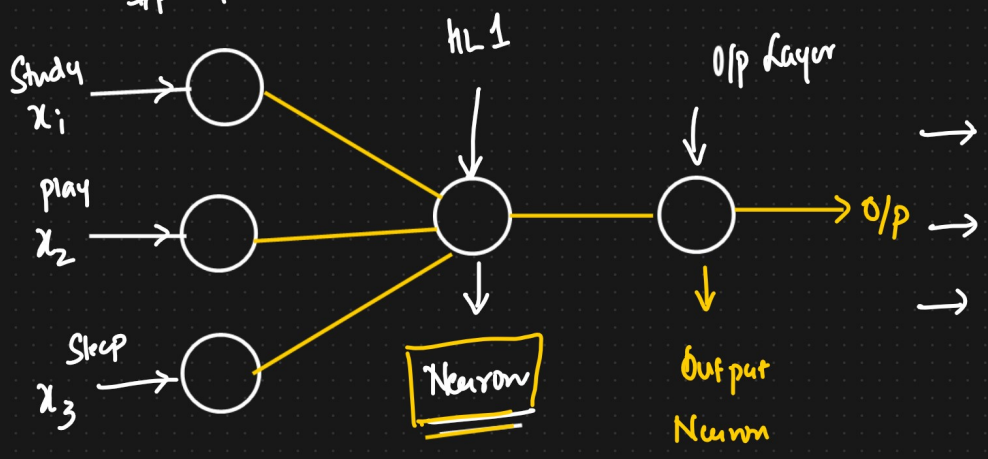
⇓

{Cumulate Revenues, Better Decisions} {RTX Titan} {RTX 3090}↑

② Hardware Advancement (NVidia) → GPU's → TRAINING THE MODEL

GPU's → COST ↓↓↓

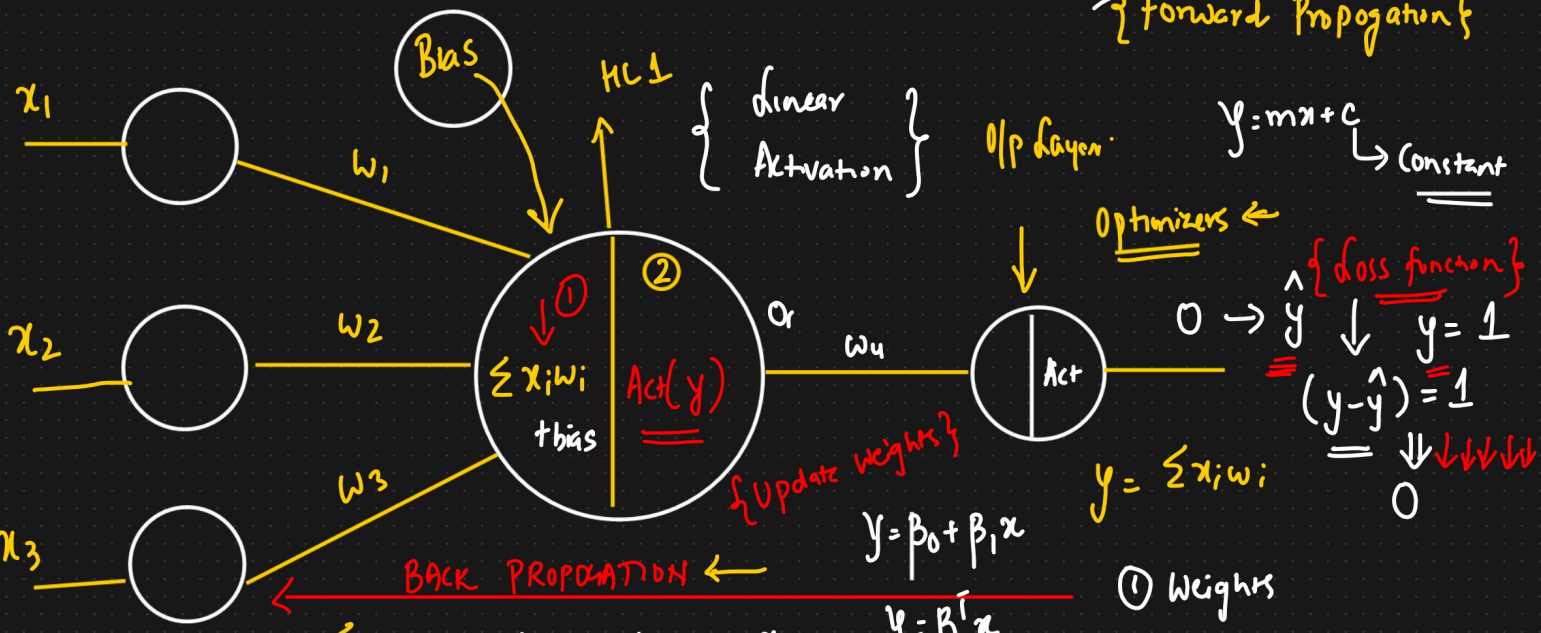
④ Perceptron {Single Layered Neural N/W}.



Binary classification

Dataset

Study	Play	Sleep	Pass/Fail
7	3	7	1
2	5	8	0
4	3	7	1



① Weights
② Neurons
③ Activations

Binary classification

↳ Sigmoid $= \frac{1}{1 + e^{-y}} = \frac{1}{1 + e^{-(\sum x_i w_i + b)}}$

0 to 1

$\geq 0.5 \rightarrow 1$

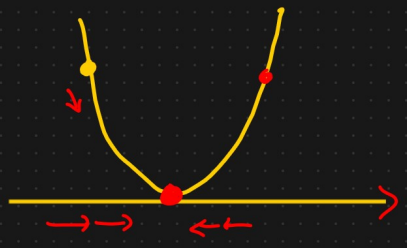
$< 0.5 \rightarrow 0$

0 or 1

① ANN
② CNN
③ RNN
④ Object Detection

GRADIENT DESCENT ⇒ kind of optimizers

$y - \hat{y}$ ⇒ minimal



Conclusion

(300 lbs)

Scribble Ink

1. I/P layers
2. Weights
3. Bias — Day 2
4. Activation function X

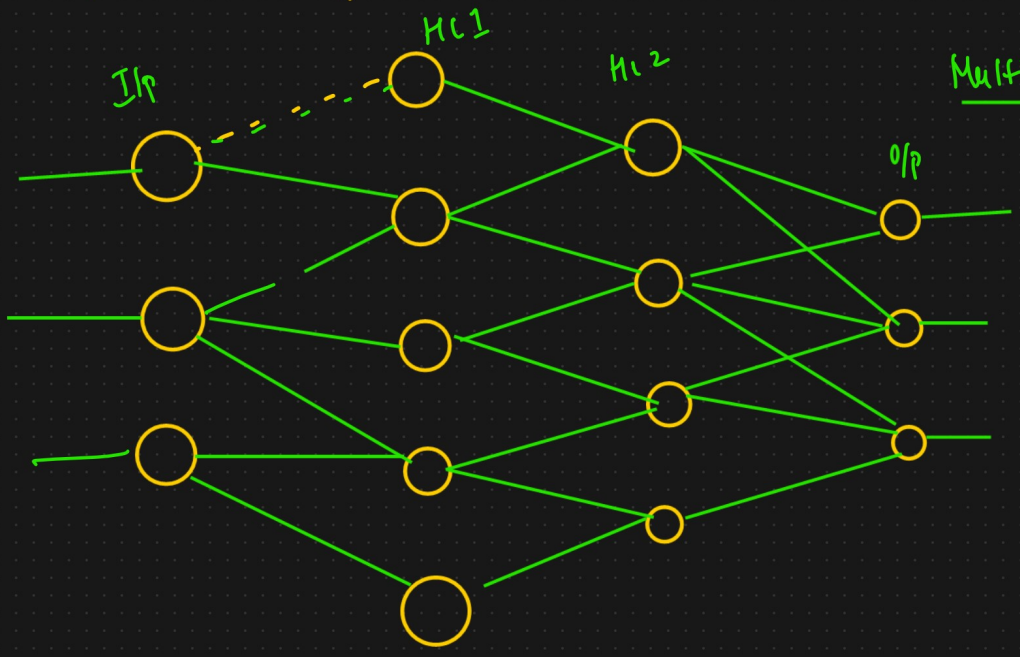
→ Forward Propogation

ANN

Chain Rule of Differentiation

5. Loss function X $\{\hat{y} - y\}$ ↓W
6. Optimizers
7. Update the weight — Day 3

→ Backward Propogation



Multi Layered N/W

1. ANN
2. CNN
3. RNN

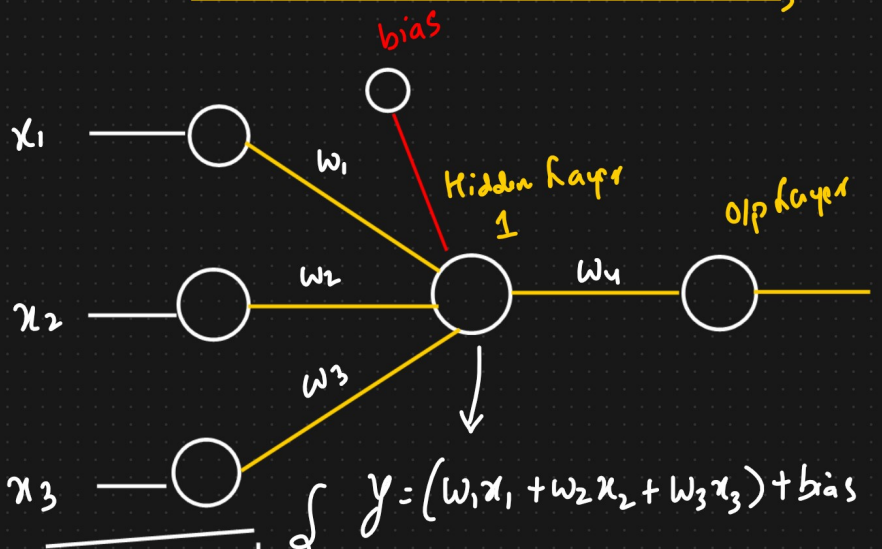
→ NLP ~ 7 days

Day 2 - Deep Learning.

Agenda

1. Forward Propogation ✓
2. Chain Rule of Derivative ✓
3. Vanishing Gradient Problem ✓
4. Loss functions ✓

Activation functions



$$y = (w_1x_1 + w_2x_2 + w_3x_3) + bias$$

$$y = w^Tx + b$$

$$z = \sigma(y) \rightarrow \text{Sigmoid Activations}$$

Loss $(y - \hat{y})$ ↓↓

⇒ Forward propogation

⇓ Non linear problems

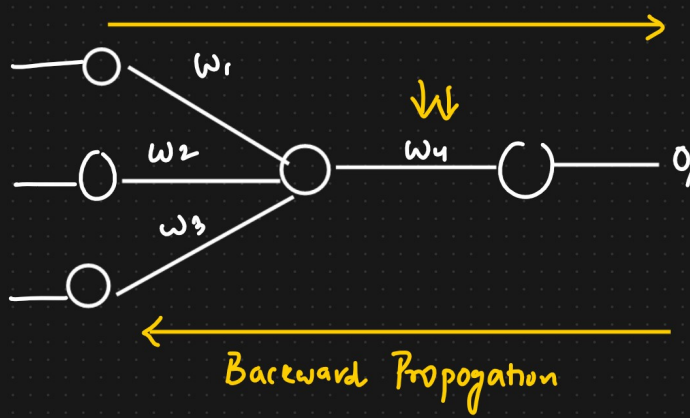
⇓ 0 or 1

$\geq 0.5 \rightarrow 1$

$< 0.5 \rightarrow 0$

② Backpropogation

1. Weight updation formula
2. Chain Rule of Differentiation ✓



L: $(y - \hat{y})$ ↓↓

$\eta = 0.01$

$\eta = 0.001$

① Weight updation formula

$$W_{new} = W_{old} - \eta \left[\frac{\partial L}{\partial W_{old}}\right]$$

Learning Rate

$\frac{\partial L}{\partial W_{old}}$ → slope

$\frac{\partial L}{\partial W_{old}}$ [-ve slope]

$W_{new} = W_{old} - \eta\,(-ve)$

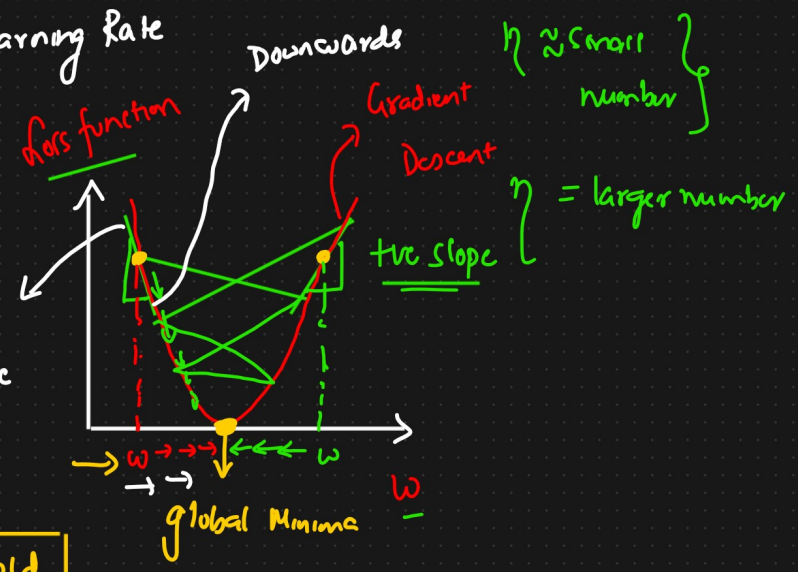
$= W_{old} + \eta\,(+ve)$

$\Downarrow$

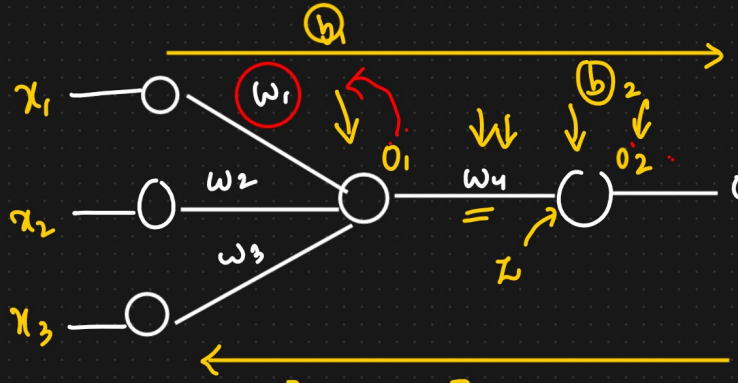
$W_{new} >> W_{old}$

$W_{new} = W_{old} - \eta\,(+ve)$

$W_{new} << W_{old}$



② Chain Rule of Differentiation



$$W_{new} = W_{old} - \eta \left[\frac{\partial L}{\partial W_{old}}\right]$$

$L = \sigma(O_1 w_4 + b)$

$L \cdot (y - \hat{y})$ ↓↓

$$b_{2_{new}} = b_{2_{old}} - \eta \frac{\partial L}{\partial b_{2_{old}}}$$

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

{Chain Rule of Derivative}

$$\frac{\partial L}{\partial W_{4_{old}}} = \frac{\partial L}{\partial O_2} * \frac{\partial O_2}{\partial W_{4_{old}}}$$

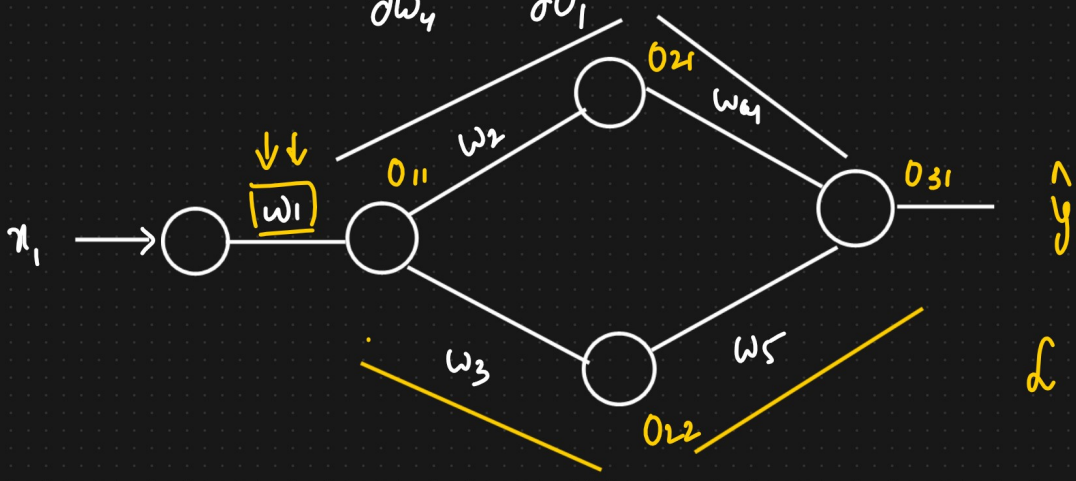
$$w_{1_{new}} = w_{1_{old}} - \eta \boxed{\frac{\partial h}{\partial w_{1_{old}}}}$$

$$\boxed{w_{2_{new}} = w_{2_{old}} - \eta \frac{\partial h}{\partial w_{2_{old}}}}$$

h → Loss

$$\boxed{\frac{\partial h}{\partial w_{1_{old}}} = \frac{\partial h}{\partial o_2} * \frac{\partial o_2}{\partial o_1} * \frac{\partial o_1}{\partial w_{1_{old}}}}$$

$$\frac{\partial o_2}{\partial w_4} * \frac{\partial w_4}{\partial o_1}$$



Optimizers

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

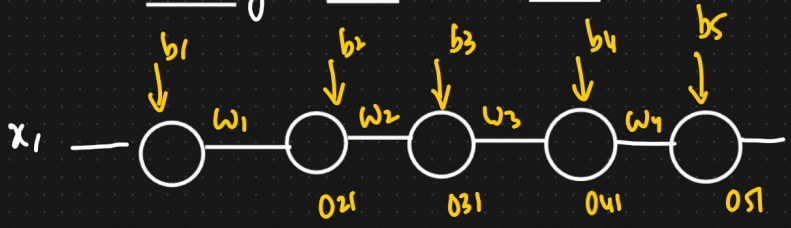
Chain Rule of Derivatives

$$\frac{\partial h}{\partial w_{1_{old}}} = \left[\frac{\partial h}{\partial o_{31}} * \frac{\partial o_{31}}{\partial o_{21}} * \frac{\partial o_{21}}{\partial o_{11}} * \frac{\partial o_{11}}{\partial w_{1_{old}}}\right]$$

$$+$$

$$\left[\frac{\partial h}{\partial o_{31}} * \frac{\partial o_{31}}{\partial o_{22}} * \frac{\partial o_{22}}{\partial o_{11}} * \frac{\partial o_{11}}{\partial w_{1_{old}}}\right]$$

③ Vanishing Gradient Problem



Sigmoid Activation

MSE

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

$$O_{51} = \sigma[(O_{41} * W_4) + b]$$

↓ Sigmoid Activation

0 to 0.25

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

$$\frac{\partial L}{\partial W_{1new}} = \frac{\partial L}{\partial O_{51}} * \frac{\partial O_{51}}{\partial O_{41}} * \frac{\partial O_{41}}{\partial O_{31}} * \frac{\partial O_{31}}{\partial O_{21}} * \frac{\partial O_{21}}{\partial W_1}$$

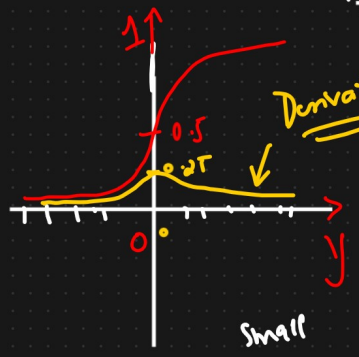
$$= 0.25 * 0.15 * 0.10 * 0.05 * 0.02$$

$$= \text{Small value}$$

Derivative:

$$0 \le \sigma(y) \le 0.25$$

$y = w^T x + b$



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

→ 0 or 1

≥ 0.5 → 1

< 0.5 → 0

Small

$$W_{new} = W_{old} - \eta \,(\text{small number})$$

$W_{new} \approx W_{old}$

⇒ Vanishing gradient Problem

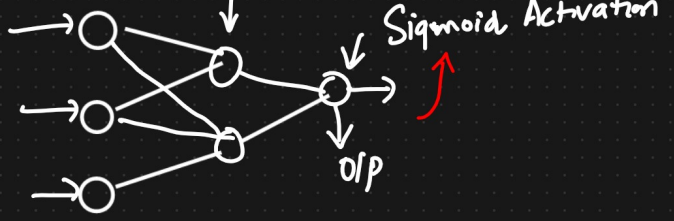
Another Activation function.

⇓ No change in weights

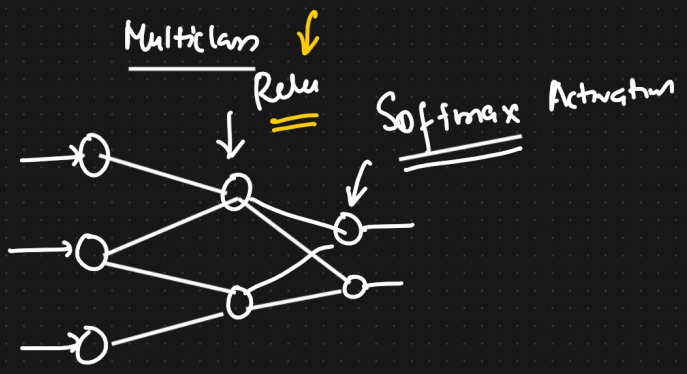
① Sigmoid
② Tanh
③ Relu
④ Leaky Relu
⑤ Prelu

Technique Which Activation fn we should Used

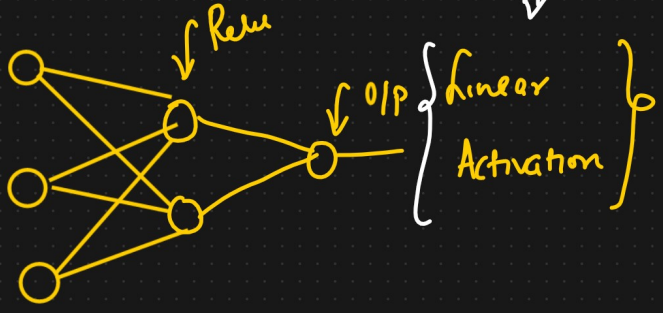
Binary classification



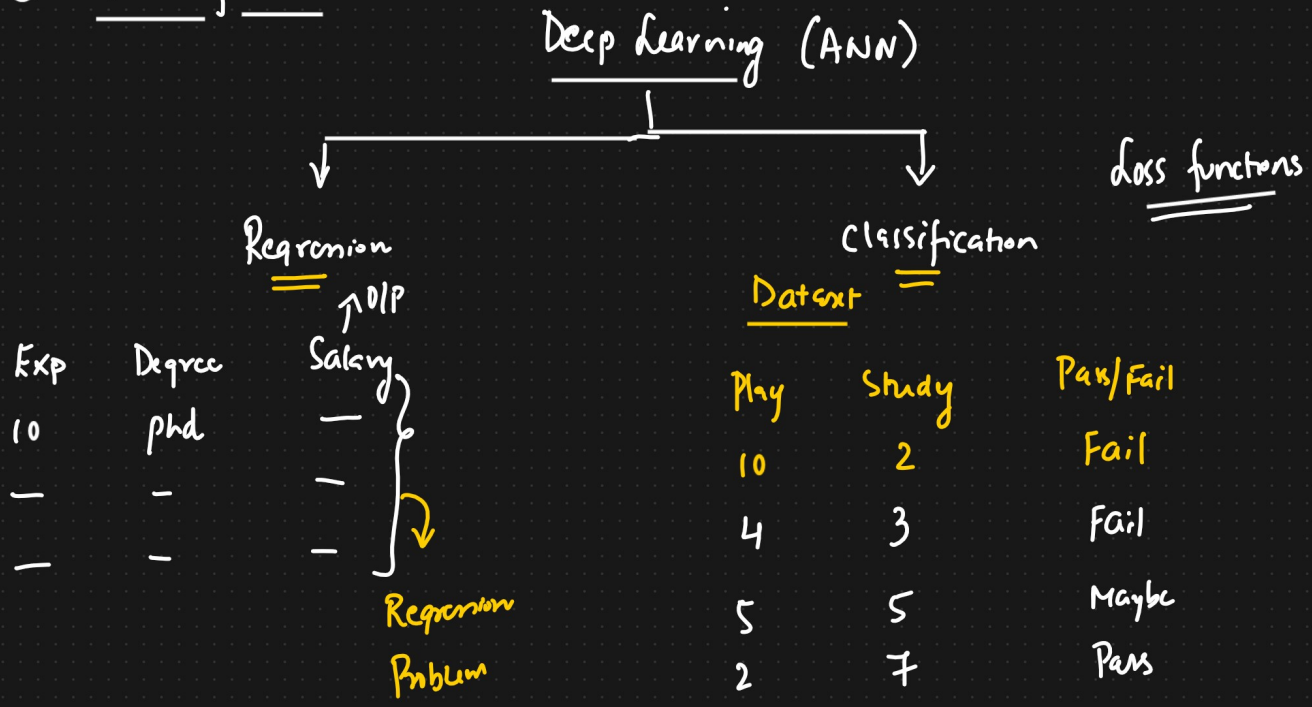
Multiclass



Regression



⑥ Loss functions

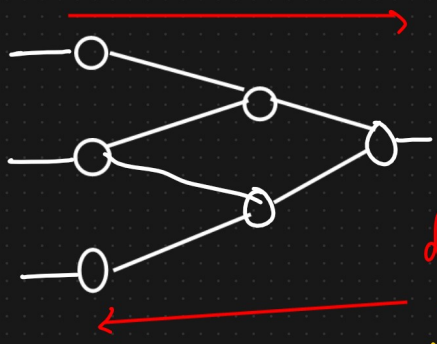


① Regression

① MSE {Mean Squared Error}
② MAE {Mean Absolute Error}
③ Huber Loss

Loss function AND
Cost Function

10 records
⇓
Dataset = 100 records



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

① Mean Squared Error (MSE)

$Loss\ function = \frac{1}{2}(y-\hat{y})^2$ ← Error

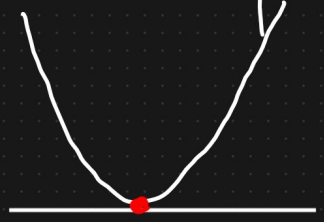
$Cost\ function = \frac{1}{2}\sum_{i=1}^{n}(y-\hat{y})^2$

⇓ Quadratic Equation

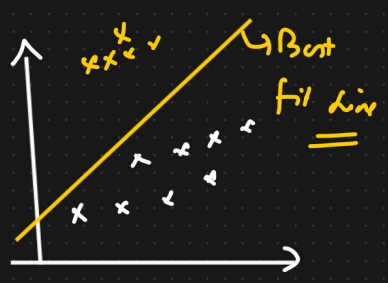
$(a-b)^2 = a^2 - 2ab + b^2$

⇓

$ax^2 + bx + c$



Penalizing Error



Advantages

① Differentiable
② It has only 1 local or Global Minima.
③ It converges faster

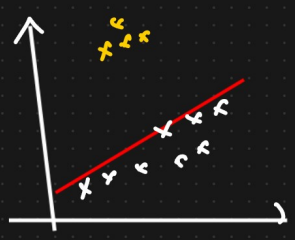
Disadvantage

① Not Robust to outliers

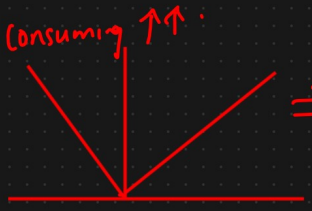
② Mean Absolute Error

$Loss\ fn = \frac{1}{2}|y-\hat{y}|$

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



① Robust to outliers ⇒ Time consuming ↑↑.



⇒ Subgradient

③ Huber Loss

① MSE ② MAE

outliers are not present

Hyperparameter

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

③ Classification

CROSS ENTROPY
- → Binary Cross Entropy → Binary classification
- → Categorical Cross Entropy → Multiclass classification.

① Binary Cross Entropy

$$\text{Loss} = -y * \log(\hat{y}) - (1-y) * \log(1-\hat{y})$$ ⇒ Logistic Regression

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

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

② Categorical CROSS ENTROPY {Multiclass Classification Problem}

f1	f2	f3	O/P
2	3	4	Good
5	6	7	Bad
8	9	10	Neutral

	Good	Bad	Neutral
Good	1	0	0
Bad	0	1	0
Neutral	0	0	1

i = Row, j = Column

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

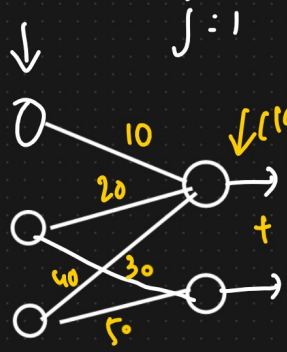
$$y_i = [y_{i1}, y_{i2}, y_{i3} \cdots y_{ic}]$$

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

$$\hat{y}_{ij} = \text{Softmax Activation} \rightarrow \text{O/p Layer}$$

$$\sigma(z) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \Rightarrow \text{Softmax Activation}$$

0.4, 0.5, 0.6



$$= \frac{e^{10}}{e^{10+20+30+40+50}} \Rightarrow$$

Conclusions

Relu, Softmax ⇒ Multiclass → Categorical Cross Entropy

Relu, Sigmoid ⇒ Binary → Binary Cross Entropy.

Linear Regression

Relu, Linear Activa → MSE, MAE, Huber Loss

Day 3 - Deep Learning.

Agenda

1. Optimizers
   - i) Gradient Descent
   - 2) SGD (Stochastic Gradient Descent)
   - 3) Mini Batch SGD
   - 4) SGD with Momentum
   - 5) Adagrad
   - 6) RMSPROP
   - 7) Adam optimizer

Batch, Epochs, Iterations

⇓

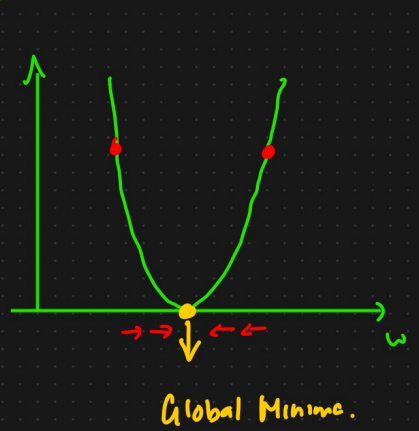
ANN

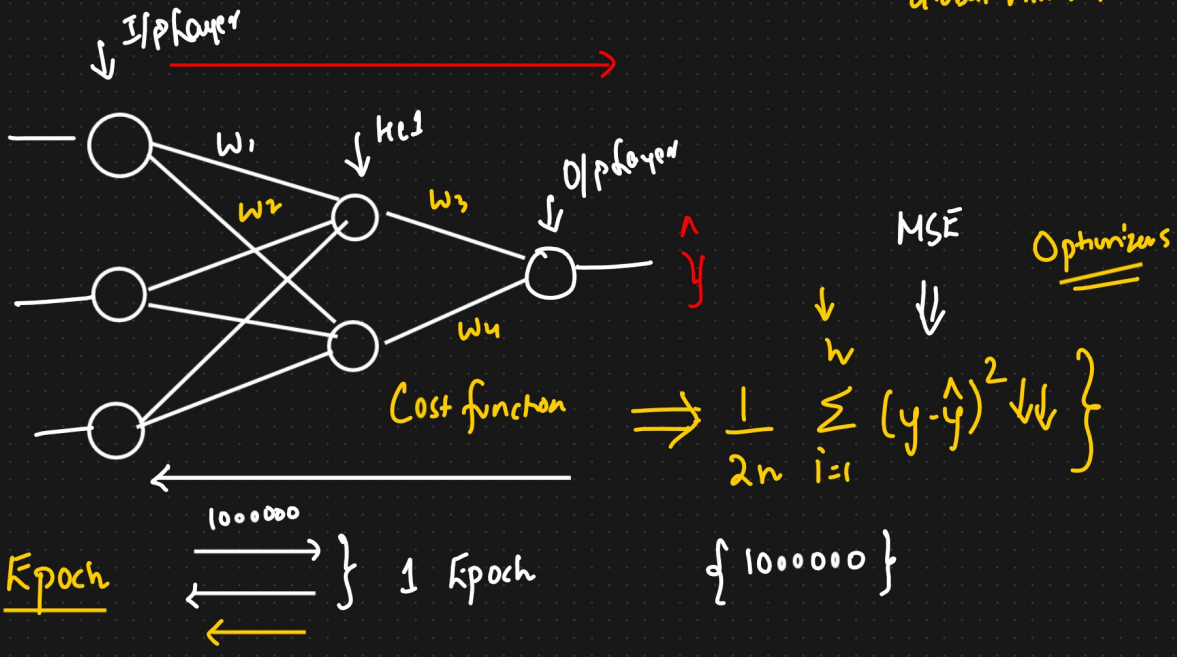
1) GRADIENT DESCENT → Optimizer

Weight Updation Formula

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

Learning Rate





MSE

Optimizers

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

{ 1000000 }

Disadvantage

① Resource Extensive {Huge RAM}

② Stochastic Gradient Descent

1000000 → 100 epochs

Epoch 1

1 record → ŷ → loss → Update weights } → Iteration 1

2 record → } Iteration 2

⋮

} Million Iteration

1000 epochs / 1 epoch

① RAM ↓↓

Disadvantage

① Convergence will be very slow

② Time Complexity will also be high

③ Mini batch SGD

1000000 batch_size = 1000

Epoch-1

1000 → } Iteration 1

1000 → } Iteration 2

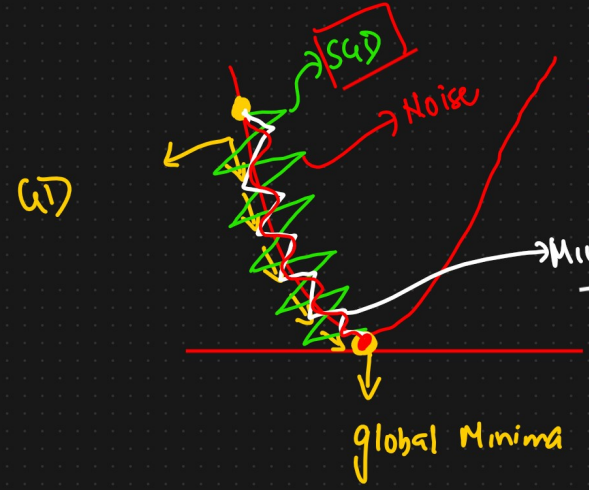
⋮

} Iteration 1000

① Resource Intensive

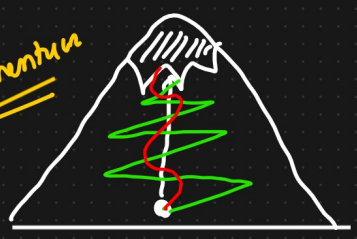
② Convergence will be better

③ Time Complexity will Improve



Remove Noise ??

Momentum



④ SGD With Momentum

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

$$b_{new} = b_{old} - \eta \frac{\partial h}{\partial b_{old}}$$

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

{Exponential Weighted Average}

⇓

Time series

⇓

ARIMA, ARMA,

Exponential Weighted Average

{Forecasting}

$t_1 \quad t_2 \quad t_3 \quad t_4 \quad \cdots \quad t_n$

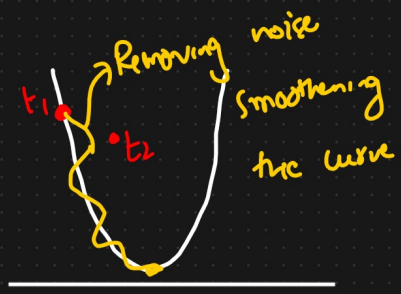
$a_1 \quad a_2 \quad a_3 \quad a_4 \quad \cdots \quad a_n$

$\beta \Rightarrow$ Hyper parameter

$\beta = 0$ to 1

⇓

0.95



$$V_{t_1} = a_1$$

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

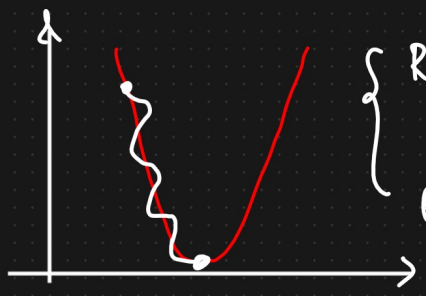
$$= (0.95) * V_{t_1} + (0.05) * a_2$$

$$V_{t_3} = \beta \times V_{t_2} + (1-\beta) * a_3$$

Exponential Weighted Avg

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

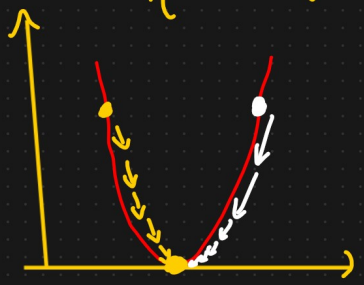
$$\boxed{V_{dw_t} = \beta \times V_{dw_{t-1}} + (1-\beta) * \frac{dL}{\partial w_{t-1}}}$$



{ Reduce the noise
Minibatch
Quicker Convergence

Recap

① Gradient Descent
② SGD
③ Mini batch SGD
④ SGD with Momentum
⑤ Adagrad → Adaptive Gradient Descent



{ fixed } ⇒ optimizer
↑
η = Learning Rate

η = fixed ⇒ adaptive ⇒ Learning Rate ⇒ Decreasing → Global Minima

Global Minima ⇓

$$W_t = W_{t-1} - \eta \frac{dL}{\partial w_{t-1}}$$

$\eta \downarrow\downarrow\downarrow$

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

ϵ ↓ Small number

$\eta \rightarrow$ ↓ Decreasing ↓↓↓

$$W_t = W_{t-1} - \eta' \frac{\partial L}{\partial w_{t-1}} \downarrow$$

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

Huge number

$W_t \approx W_{t-1}$

t=1	t=2	t=3
$\eta = 0.01$	$\eta = 0.005$	$\eta = 0.002$

⑥ Adadelta and Rmsprop

Exponential Weighted Average ⇓

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

$Sdw = 0$

$$Sdw_t = \beta\, Sdw_{t-1} + (1-\beta)\left(\frac{\partial L}{\partial w_{t-1}}\right)^2$$

$\beta = 0.95$

$$Sdw_t = (0.95)\, Sdw_{t-1} + (0.05)\left(\frac{\partial L}{\partial w_{t-1}}\right)^2$$

(F) Adam Optimizer (Best Optimizer)

Momentum + RMSprop (Adaptive Learning Rate)

1. Smoothening
2. Learning Rate Adaptive

$Vdw = 0$ $Vdb = 0$ $Sdw = 0$ $Sdb = 0$

b)

$$w_t = w_{t-1} - \eta' Vdw$$

$$b_t = w_{b\,t-1} - \eta' Vdb$$

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

$$Vdw_t = \beta \times Vdw_{t-1} + (1-\beta)\frac{\partial L}{\partial w_{t-1}}$$

$$Vdb_t = \beta \times Vdb_{t-1} + (1-\beta)\frac{\partial L}{\partial b_{t-1}}$$