Father of Soft computing : Lotfi A. Zadeh Sir.
Fuzzy logic

*Genetic Algorithm → John Holland

* Neural Network → Artificial NN : Geoffrey Hinton.
DL

Convolutional NN : Yann LeCun.

*Godfathers of AI → Yoshua Bengio
Geoffrey Hinton
Yann LeCun.

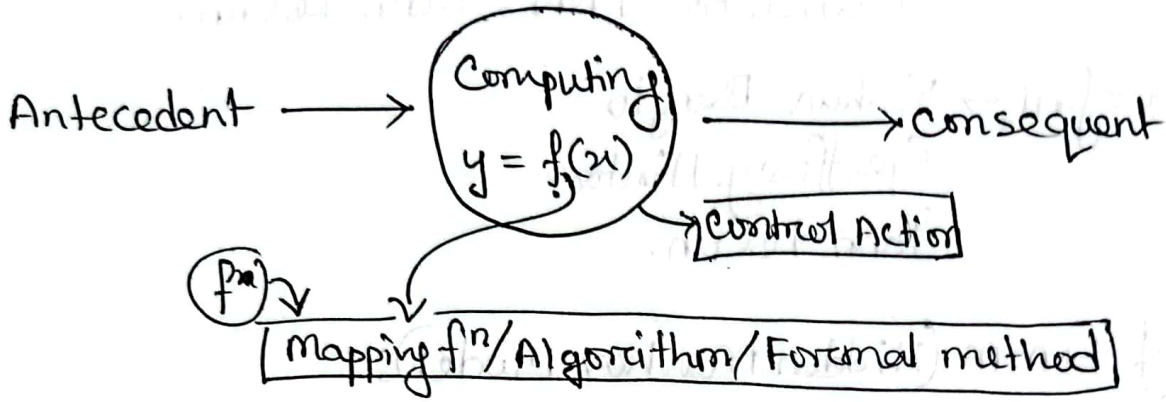
*Father of HMM (Hidden Markov Model):

**Donald Knuth.

*Computation : Input- নিয়ে operators দিয়ে process করে outcome বের করা.

$f(x) \rightarrow y$.

*Control Unit :



*Computing :

→ process/act of calculation.
→ action of mathematical calculation
→ activity that uses Computers.
→ development of hw and sw.
→ critical, integral component

* Brain	vs	Computer
1. 10^{10} neurons		10^8 transistors
2. Element size 10^{-6} m		10^{-6} m
3. Energy 30W		30W (CPU)
4. Speed 10^2 Hz		10^{12} Hz
5. Parallel computation Distributed		Serial Centralized.
6. Efficiency 10^{-16} J.		10^{-6} J.
7. Fault Tolerant	yes	no
8. Learns	yes	a little

⊞ Accuracy - overall performance
⊞ Precision - কতটা precise - প্রতিবার accuracy কমছে না বাড়ছে।

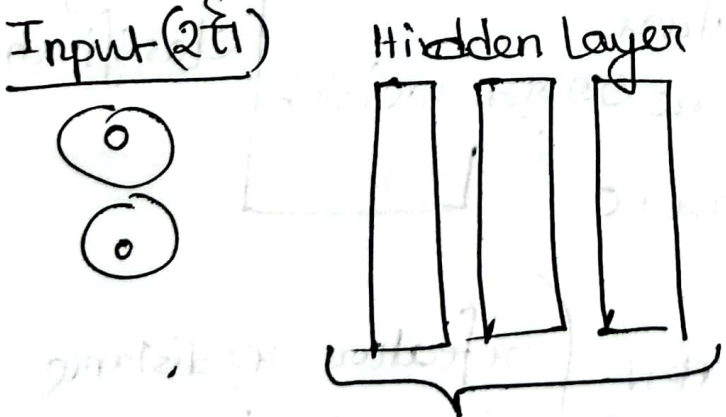
⊞ ML → data কম
⊞ DL → more data

(*) Hard computing (vs) Soft computing (Computational intelligence)

1. Precise result → imprecision, approximate, uncertain results
2. Unambiguous → ambiguous control action
3. Formally defined → adaptive
4. Computation time more → less.
5. Crisp sw, binary logic, numerical sys → based on approximation and dispositional (not fixed)
6. sequential computation → parallel.
7. works on exact data → on ambigous & noisy data.
8. Uses two-valued logic → multivalued logic
9. settled computing → incorporates randomness.
10. requires programs to be written → emerges its own programs
11. deterministic nature → stochastic (random)

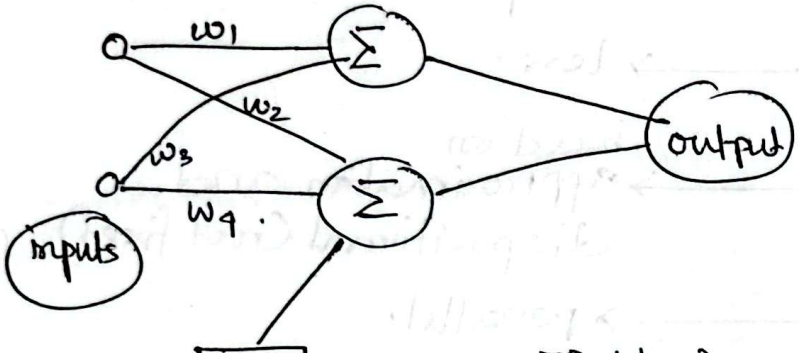
Neural Network (works great on atleast- 10K data)

* Inspired by brain neurons.



Shallow → <3. | layer GPU create करो.
Deep → >3. |

* আমাদের target Hardware-এ problem dot product-এ নিয়ে আসা as matrix, তাহলে dot product-এ আনলে super fast হবে।



Training Set - trains
Test set — tests
Validation set - ↓ Error - underfitting না হওয়া control করে

* Hyper parameters: fixed [Batch size, Epoch size, iteration, learning rate]
* Parameters (weight, bias): change করা possible; updates, or is learned

connection line between input-hidden and hidden-output.

Controls orientation of decision boundary. (aka hyperplane)

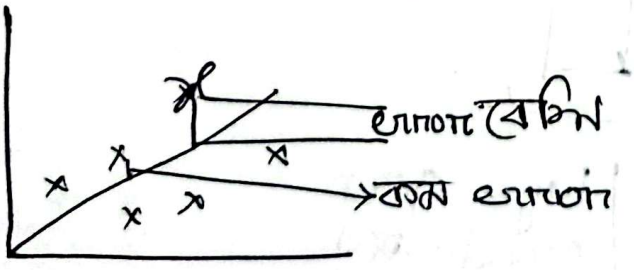
Controls location

→ suppose phone কিনতে গেছি iphone নিব না android confused, সাথে iphone wala friend বলল iphone নিতে। Iphone wala friend is bias.

* image থেকে pixels-এ feature বের করি
20 x 20 pixels ⟶ 400টি feature
যতগুলা feature ততগুলা weight.

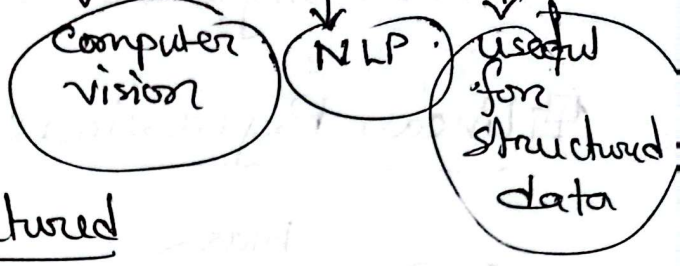
Binary classification.

* Linear regression: $y = mx + c$



x feature-এর distance from line is error.

Supervised learning - 2 CNN, RNN(recurrent), Standard NN and Hybrid/customNN use হয়।



(*) Structured (vs) Unstructured

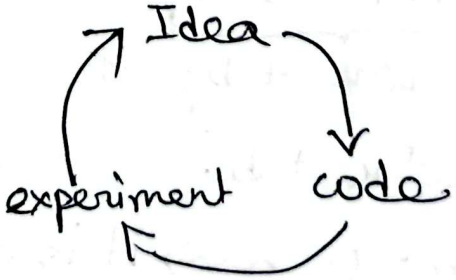
1. Database, tables — Audio, text, images
2. More money (companies big data prediction)

~~(*) Algorithms:~~
~~→ creative algorithms.~~

(*) ReLU, SIGMOID, TanH
↓
better than sigmoid.
(as helps with vanishing gradient problem)

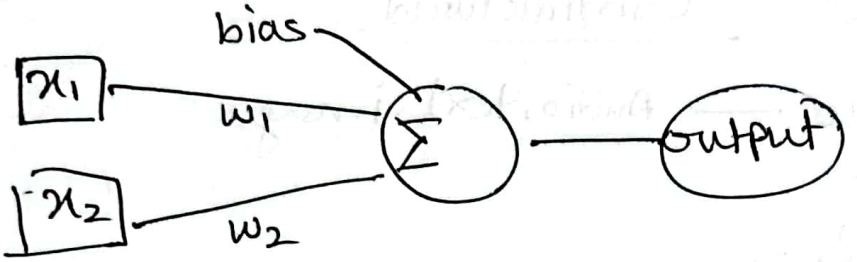
~~(*)~~ (*) Why DL? → society তে অনেক data
→ fast algo generate হচ্ছে
→ GPU works faster.

(*) Iterative training :



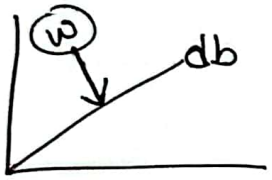
Logistic Regression: nonlinear data নিয়ে কাজ করে
(একটা single unit neuron network (একটা node থাকবে)

Linear Regression: works with Linear data.



* Linear sum equation → $w_1x_1 + w_2x_2 + \text{bias} := \Sigma$.

* weight is perpendicular to db; $w \perp db$.



Decision boundary's equation:

(*) In vector → $[w_1, w_2] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

So, db-র equation = $w_1x_1 + w_2x_2$

$$g(x) = \underbrace{w_1x_1 + w_2x_2}_{wx} + b.$$

$$= wx + b.$$

$$= w^Tx + b.$$

if $w^Tx + b = 0$ ⇝ it is decision boundary (if x on db then যেকোনো class-এর হয়)

" $w^Tx + b > 0$ ..... $x \in$ Class 1

" $w^Tx + b < 0$ ..... $x \in$ Class 2.

↓ Threshold value.

Facts of logistic Regression.

↳ supervised.
↳ output between 0-1.
↳ goal is to minimize error btw. prediction & actual data.

(VS) linear regression.
↳ data linearly & seperated
but ~~LR~~ Log R-1 nonLinearly seperated.

Maths of Logistic Regression:

Let y = actual data
$\hat{y}$ = predicted data

$\hat{y} = P(y = 1 | x)$
↳ probability of x feature of being class 1.

(*) Model always returns probability.

Mathematical parameters:

1 (*) Input → $x \in R^{n_x}$ (n_x ∈ 1 feature)
↳ real no.

2 (*) Training Label → $y \in 0, 1$
↳ actual value ↳ labels.

3. Feature সংখ্যক weight আছে — so $w \in R^{n_x}$

4. Bias → $b \in R$

So, input, weight, bias সব Real numbers.

5. Output → $\hat{y} = \sigma \cdot (w^T x + b)$

sigmoid

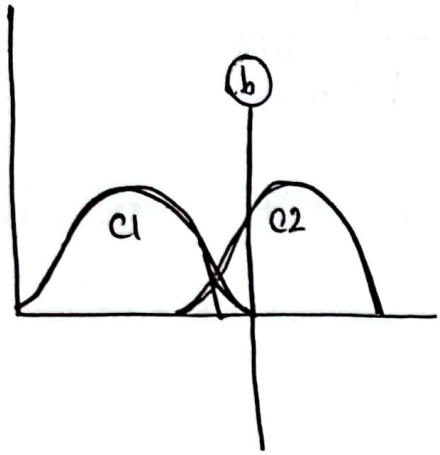
6. Sigmoid function : $s = \sigma(w^T x + b) = \sigma(z) = \frac{1}{1+e^{-z}}$

input

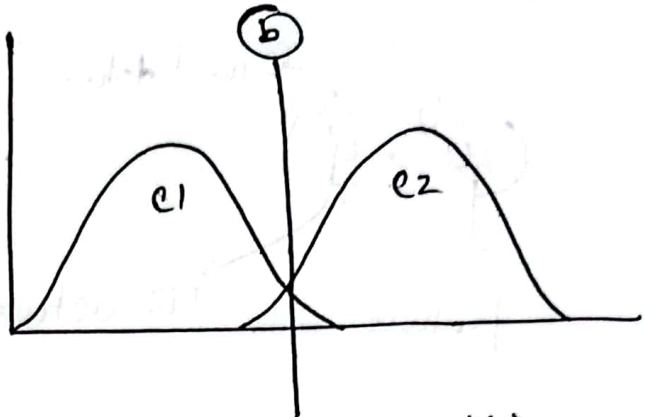
7. $w_1 x_1 + b$; b বাড়লে output value বাড়ে

*

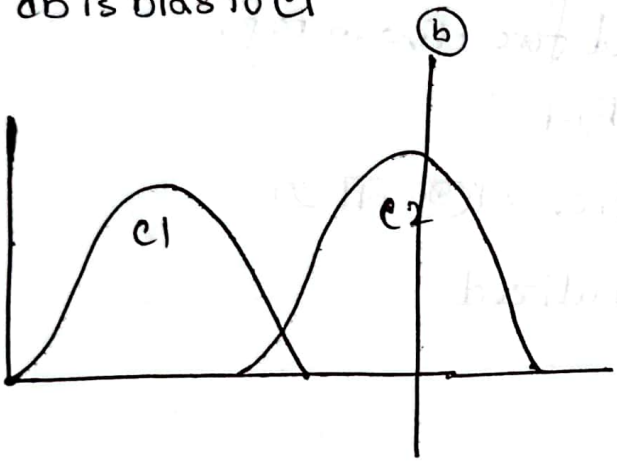
bias positions in graph meanings:



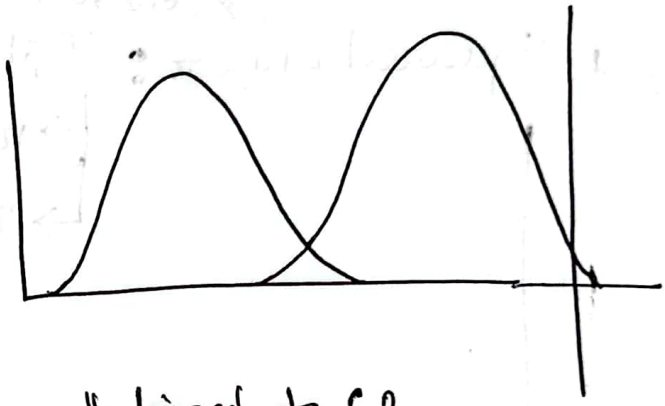
db is bias to C1



better position



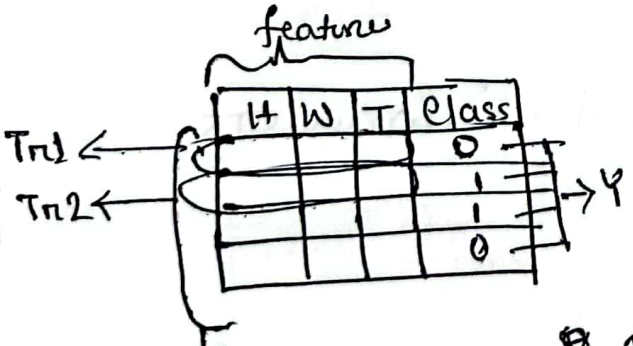
dB is bias to C2



db biased to C2

Loss fn : $(\hat{y} - y)$ = Loss fn ⇝ gap between predicted and actual data (single training example - error).

Cost fn : full dataset → loss - → avg.



N data.

$\hat{Y} - Y$ = loss fn of Trl

N data loss fn = cost fn.

(ক) Cost fn → $\hat{y}^{(i)} = \sigma(w^T x^{(i)} + b)$, where

$$\sigma(z^{(i)}) = \frac{1}{1 + e^{z^{(i)}}}$$

$(x^{(i)}, y^{(i)})$ → $y^{(i)}$ → no. of data.

$x^{(i)}$ → feature

$y^{(i)}$ → তার actual Label

⊞ Loss fn :

1. Squared Error : → commonly used for linear reg. করার কারণ
   - → negative যাতে না হয়
   - → To peralized

$$L(\hat{y}^{(i)}, y^{(i)}) = \frac{1}{2}(\hat{y}^{(i)} - y^{(i)})^2$$

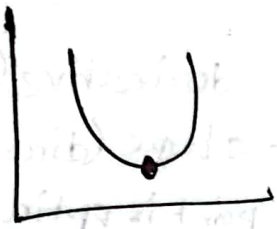
→ square করছে negative value avoid করতে.

→ 2 করছে, square-এর তার value-তে extra 2 কাটাকাটি করে হিসাব জন্য.

But this stucks at local minima or maxima (not optimized)

Optimization ⟶ getting the global maxima or minima; maximum/minimum fn.

(*) Convex graph is better ~~at~~ as minima or maxima একটাই



(*) Non-convex → local → অনেকগুলো সুযোগ আছে ~~times~~.

2. Cross Entropy loss :-

$$L(\hat{y}^{(i)}, y^{(i)}) = -\left(y^{(i)} \log(\hat{y}^{(i)}) + (1-y^{(i)}) \log(1-\hat{y}^{(i)})\right)$$

If $y^{(i)} = 1$; $L(\hat{y}^{(i)}, y^{(i)}) = -\log(\hat{y}^{(i)})$ where $\hat{y}^{(i)}$ and $\log(\hat{y}^{(i)})$ near to 1.

আর 0 - হলে 0 -র কাছে।

cost fn → avg of loss fn of the entire training set.

$$J(w,b) = \frac{1}{m} \sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$$

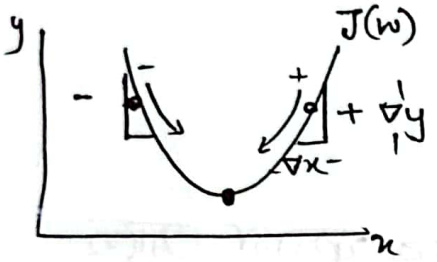
$$= -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log(\hat{y}^{(i)}) + (1-y^{(i)}) \log(1-\hat{y}^{(i)})\right]$$

We want to find, w and b · to minimize cost fn $J(w,b)$

Gaviflux

Gradient Descent :-

→ cost fn is convex.
→ initialize w and b with random numbers (0,0 → incase of logistic reg). তারপর try to IMPROVE.
→ যেহেতু fn convex – initialize যেই। কিনেই হোক global optimal or near to it যাবেই।



Gradient – derivative (slope of a fn at a point)
Descent – slope (direction পাই)
if fn = 0 ; point is optimal.

For update we use, $w(t+1) = w(t) - \alpha \frac{\partial J(w,b)}{\partial w}$ ← partial derivative

$b = b - \alpha \frac{\partial J(w,b)}{\partial b}$

(α) → learning rate. (how bigger step we choose at each itr).

Necessity of GD:

Solves time problem.
optimal solution পাওয়া

"Premature optimization is the root of all evil" – Donald knuth.

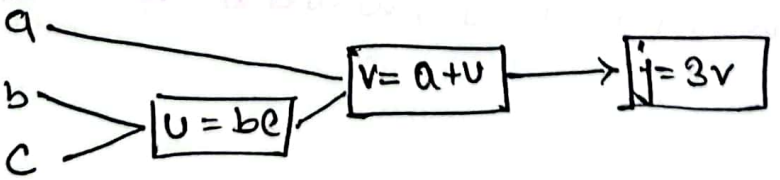
✱ **Computational Graph:** why we need it? → to represent full NN + explains why it is organized that way

✱ **Forward pass:** Computes output

✱ **Backward pass:** ~~der~~ " derivatives + gradient to update w and b.

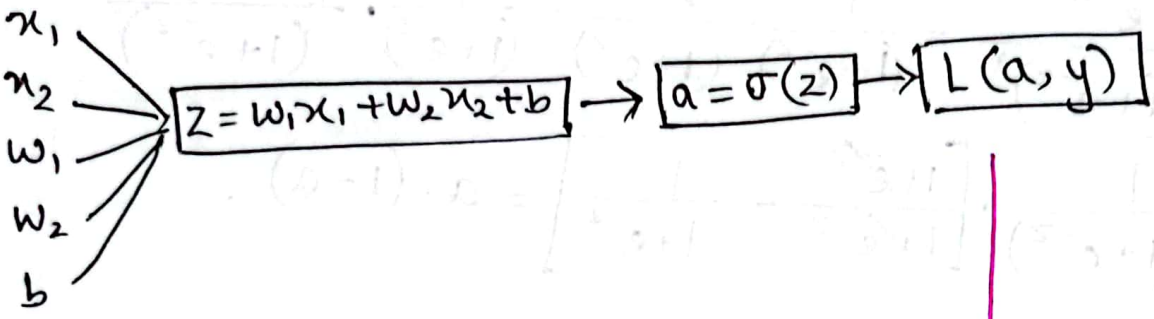
$J(a,b,c) = 3(a+bc)$.

① $u = bc$
② $v = a + bu$
③ $J = 3v$



Forward:

$$z = w^T x + b$$
$$\hat{y} = a = \sigma(z)$$
$$L(a, y) = -(y \log(a) + (1-y) \log(1-a)).$$



Backward:

$$da = \frac{\partial L(a,y)}{\partial a}$$ ↳ deriv. of Loss with respect to a.

$$dz = \frac{\partial L(a,y)}{\partial z} = \frac{\partial L}{\partial z}$$ deriv of L with z.

for, $\frac{\partial L(a,y)}{\partial a} = \frac{d}{da}[y \log(a) + (1-y) \log(1-a)]$ [∵ without - sign]

$= y \frac{d}{da}[\ln a] + (1-y) \cdot \frac{d}{da}[\ln(1-a)]$

$= y \cdot \frac{1}{a} \cdot \frac{da}{da} + (1-y) \cdot \frac{1}{(1-a)} \cdot \frac{d}{da}(1-a)$

$= \frac{y}{a} + \frac{(1-y)}{(1-a)} \left(\frac{d}{da} - \frac{da}{da}\right)$

$= \frac{y}{a} + \frac{(1-y)}{(1-a)} \times (0-1)$

$= \frac{y}{a} + \frac{(y-1)}{(1-a)}$

adding '-' sign → $\boxed{-\frac{y}{a} + \frac{(1-y)}{(1-a)}}$

a - কে sigmoid - এ পরিণত z - এর সাপেক্ষে derive করব।

$$\frac{\partial a}{\partial z} = \frac{\partial \sigma(z)}{\partial z} = \frac{\partial}{\partial z}\left[\frac{1}{1+e^{-z}}\right]$$

$$= \frac{e^{-z}}{(1+e^{-z})^2} = \frac{1}{(1+e^{-z})} \cdot \frac{e^{-z}}{(1+e^{-z})} = \frac{1}{(1+e^{-z})} \cdot \frac{(1+e^{-z})-1}{(1+e^{-z})}$$

$$= \frac{1}{(1+e^{-z})} \cdot \left[\frac{1+e^{-z}}{1+e^{-z}} - \frac{1}{1+e^{-z}}\right] = a \cdot (1-a).$$

*

Applying chain Rule → $\frac{\partial L}{\partial a} \cdot \frac{\partial a}{\partial z} = \frac{\partial L}{\partial z}$

$\downarrow$

$dz = (a - y)$

~~$\frac{\partial w_1}{\partial} \neq \frac{\partial w_1}{\partial} = \frac{\partial L}{\partial w_1} \cdot \frac{\partial w_1}{\partial z}$~~

$$\partial w_1 = \frac{\partial L}{\partial z} \cdot \frac{\partial z}{\partial w_1}$$

$$= dz \cdot \frac{\partial}{\partial w_1}(w_1 x_1 + w_2 x_2 + b)$$

$$= dz \cdot x_1$$

So, $\partial w_1 = dz\, x_1$

$\partial w_2 = dz\, x_2$

$\partial b = \cdot dz \cdot$

update: $w_1 = w_1 - \alpha * dw_1$

$w_2 = w_2 - \alpha * dw_2$

$b = b - \alpha db.$

Logistic regression GD on m examples.

$J=0$; dw1=0; dw2=0; db=0;
w1=0; w2=0; b=0;

modified (1 for loop).

→ dw = np.zeros((nx, 1))

for i = 1 to m

#Forward
$z(i) = w1 * x1(i) + w2 * x2(i) + b$.
$a(i) = \text{sigmoid}(z(i))$.
$J += (y(i) * \log(a(i)) + (1 - y(i)) \log(a(i)))$.

#Backward.
$dz(i) = a(i) - y(i)$
dw1 += dz(i) * x1(i)
dw2 += dz(i) * x2(i)
db += dz(i)

→ dw += x(i) * dz(i)

J /= m
dw1 /= m
dw2 /= m
db /= m

→ dw /= m

#Gradient.
w1 = w1 − α * dw1
w2 = w1 − α * dw2
b = b − α * db.

Problem:
① 2 for loops. for 1 step of GD.
② less efficient.

Soln:
① Vectorization.
② do not use for loop

Still loop প্রয়োজন

Vectorization :- (LR).

$z^{(1)} = w^T x^1 + b$ ..... $z^m = w^T x^m + b$

$a^1 = \sigma(z^1)$ ..... $a^m = \sigma(z^m)$.

input matrix

$$X = \begin{bmatrix} \vdots & \vdots & & \vdots \\ x^1 & x^2 & \cdots & x^m \\ \vdots & \vdots & & \vdots \end{bmatrix} \text{ while } R^{n_x \times m}$$

$$\begin{bmatrix} w^T \end{bmatrix} \begin{bmatrix} \vdots & \vdots & & \vdots \\ x_1 & x^2 & \cdots & x^m \\ \vdots & \vdots & & \vdots \end{bmatrix}$$

broadcasting.

① So, $[z^1\ z^2\ z^3 \ldots z^m] = w^T X + [b\ b\ b \ldots b]$ → Z = np.dot(w.T, X) + b

$= [w^T x^1 + b \quad w^T x^2 + b \quad \ldots \quad w^T x^m + b]$

② $A = [a^1\ a^2\ a^3 \ldots a^m] = \sigma(Z)$

③ Gradient- compute:

$dz^1 = a^1 - y^1$. $dz^2 = a^2 - y^2$ ..... $dz^m = a^m - y^m$.

$dZ = [dz^1\ dz^2 \ldots dz^m]$

$A = [a^1\ a^2\ a^3 \ldots a^m]$ $Y = [y^1\ y^2 \ldots y^m]$

$dZ = A - Y$

$= [(a^1 - y^1) \quad (a^2 - y^2) \quad \ldots \quad (a^m - y^m)]$

Vectorization implementation:

$Z = w^T X + b$.

$Z = np.dot(w.T, X) + b$.

$A = \sigma(Z)$

$dZ = A - Y$.

$dw = \frac{1}{m} X . dZ^T$

$db = \frac{1}{m} dz(i)$

$db = \frac{1}{m} np.sum(dZ)$

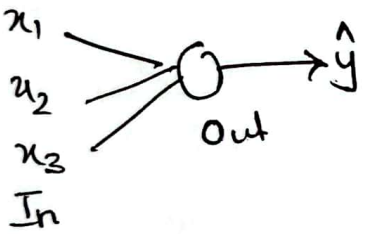
update

$w = w - \alpha * dw$

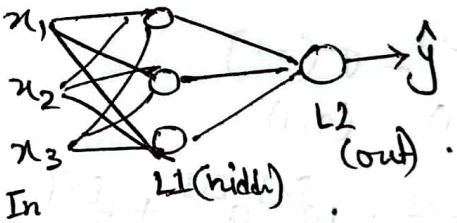
$b = b - \alpha * db$

~~Comparison of brain and NN~~

logistic single node.

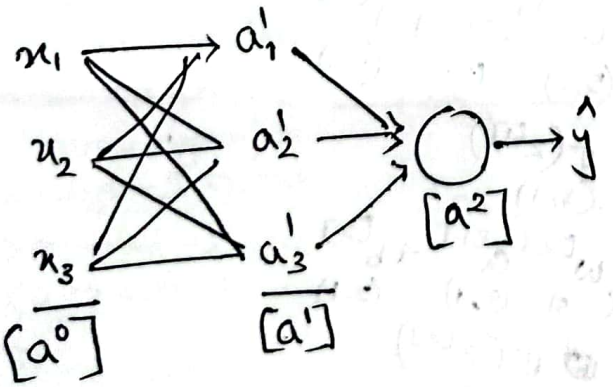


NN.



Total connection = input × hidden layer.

Let $a^{[layer]}_{node\ no.}$

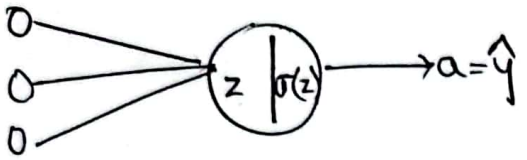


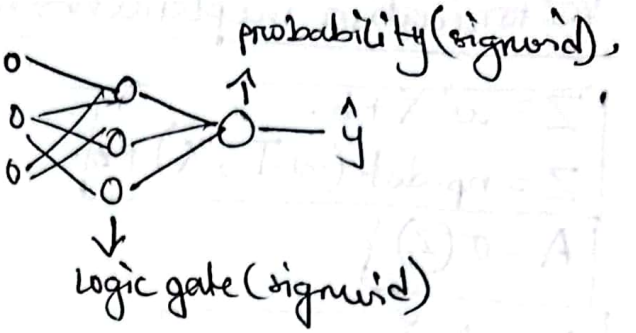
do not count input layer

↳ 2 layered neural network.

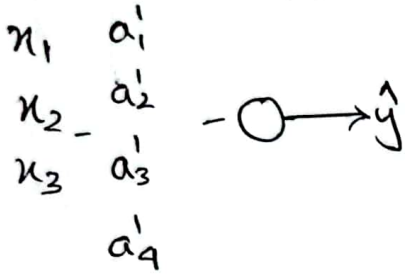
Gaviflux

Neural Net





Neural Network Representation



$$z^{[1]} = \begin{bmatrix} z_1^1 \\ \vdots \\ z_4^1 \end{bmatrix} = \begin{bmatrix} \cdots & w_1^T & \cdots \\ & \vdots & \\ \cdots & w_4^T & \cdots \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1^1 \\ \vdots \\ b_4^1 \end{bmatrix}$$

$z^{[1]} \in \mathbb{R}^{4\times1}$ $\quad w^1 \in \mathbb{R}^{4\times3}$ $\quad x \in \mathbb{R}^{3\times1}$ $\quad b \in \mathbb{R}^{4\times1}$

So,

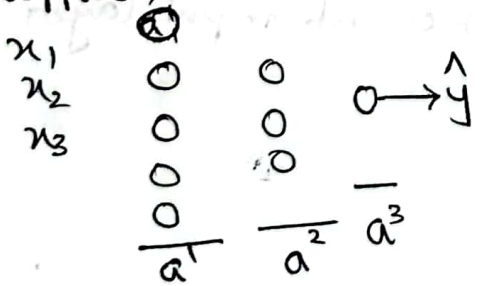
$$z^{[1]}_{(4,1)} = w^{[1]}_{(4,3)} x_{(3,1)} + b^{[1]}_{(4,1)}$$

$$a^{[1]}_{(4,1)} = \sigma(z^1)_{(4,1)}$$

$$z^{[2]}_{(1,1)} = w^{[2]}_{(1,4)} a^{[1]}_{(4,1)} + b^{[2]}_{(1,1)}$$

$$a^{[2]}_{(1,1)} = \sigma(z^{[2]})_{(1,1)}$$

Suppose



$$z^{[1]}_{(5,1)} = w^{[1]}_{(5,3)} \cdot x_{(3,1)} + b^{[1]}_{(5,1)}$$

$$a^{[1]}_{(5,1)} = \sigma(z^{[1]})_{(5,1)}$$

$$z^{[2]}_{(3,1)} = w^{[2]}_{(3,5)} a^{[1]}_{(5,1)} + b^{[2]}_{(3,1)}$$

$$a^{[2]}_{(3,1)} = \sigma(z^{[2]})_{(3,1)}$$

$$z^{[3]}_{(1,1)} = w^{[3]}_{(1,3)} a^{[2]}_{(3,1)} + b^{[3]}_{(1,1)}$$

$$a^{[3]}_{(1,1)} = \sigma(z^{[3]})_{(1,1)}$$

Justification for vectorized implementation.

$z^{[1](1)} = w^{[1]} x^{(1)} + b^{[1]}$ $\quad z^{[1](2)} = w^{[1]} x^{(2)} + b^{[1]}$. $\quad z^{[1](3)} = w^{[1]} x^{(3)} + b^{[1]}$

$$w^{[1]} = \begin{bmatrix} - & - & \cdot \\ - & - & \cdot \\ - & - & \cdot \\ - & - & \cdot \end{bmatrix} \quad w^{[1]} x^{(1)} = \begin{bmatrix} \square \\ \square \\ \square \\ \square \end{bmatrix} \quad w^{[1]} x^{(2)} = \begin{bmatrix} \Delta \\ \Delta \\ \Delta \\ \Delta \end{bmatrix} \quad w^{[1]} x^{(3)} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$w^{[1]} \times \begin{bmatrix} x^1 & x^2 & x^3 \end{bmatrix} = \begin{bmatrix} \square & \Delta & 0 \\ \square & \Delta & 0 \\ \square & \Delta & 0 \\ \square & \Delta & 0 \end{bmatrix} = \begin{bmatrix} z^{[1](1)} & z^{[1](2)} & z^{[1](3)} \end{bmatrix} = Z^{[1]}$$

$$Z^{[1]} = W^{[1]} X + b^{[1]}$$

Activation function:

NN without an activation fn or with a linear act. fn is just a linear regression model.

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial a} \cdot \frac{\partial a}{\partial z} \cdot \frac{\partial z}{\partial w_i} \quad (L, a, z, w).$$

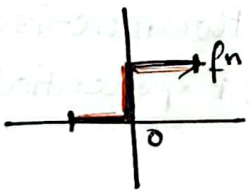
$z = w^T x + b.$

$a = g(z).$

↳ activation fn (may be linear or non linear).

① Binary Step Fn :- {0, 1}

* Threshold based.
* above or below input হলে neuron active হয়ে same signal next layer-এ পাঠায়
* $y > 0 \rightarrow out = 1$
  $y < 0 \rightarrow out = 0.$



Γ → symbol.

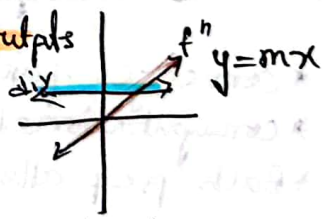
Dis:

① multivalue outputs - not allowed

② derivative - 0 আসে of constant does not represent derivative LR - so hidden layer-এ use করি না।

② Linear Act. Fn: $(-\infty, \infty)$

* Allows multivalue outputs



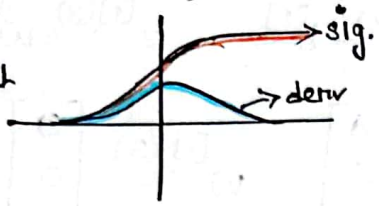
But

1. No backpropagation, w, b no update.
2. Linear-র combination is always linear — so NN will turn into one LAYER.

Gaviflux

3. Sigmoid: $\sigma = \left(\frac{1}{1+e^{-z}}\right)$ (0,1)

* Non linear
* Smooth gradient
* [0 → 1] range → output.



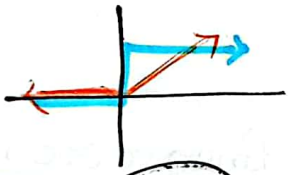
* prevents jumps.
* back propagation allowed.

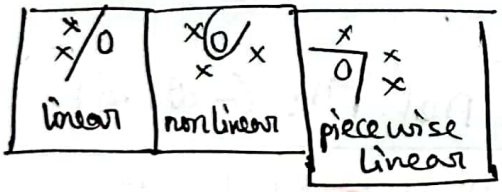
Dis:

1. Vanishing Gradient: For High or very Low value — no change, derv converges to 0.
2. Certain point → গ্রাফ flat.
3. refuses to learn further + too slow.
4. Stuck at local minima....
5. Output not ZERO CENTERED. - so সবার sign same হবে, (but real life data - neg ও হয়)
6. Computationally Expensive.

5. Rectified Linear Unit (ReLU) [0, ∞)

* Solves ✓ VG.
  ↳ by piece wise linear.





max(0, x) range.

* converges quickly
* computationally efficient
* Back prop allowed.

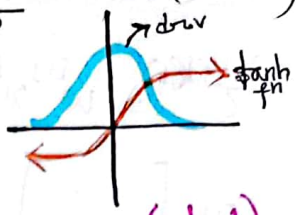
Dis:

1. Dropout problem: Derivative 0 না হলেও node নিজে থেকে off হয়ে যায়
2. Dying ReLU : when input reaches 0 or neg, grad fn 0 হয়ে যায় → back propagation off হয়ে যায় thus.
3. Regularization problem: কোন কোন node off হবে not in my hand.

But dropout-এর জন্য calculation speed বাড়ে, Load কমে।

4. Tanh hyperbolic Tangent: $\tanh\left(\frac{e^{-x}+e^{x}}{e^{-x}+e^{x}}\right)$

* Sigmoid-এর চেয়ে কিছু মানায়ের → range of output is now neg-pos; as zero centered.



(-1, 1)

* Derivative is steeper.
* Wider range + fast learning.
* Back propagation allowed

Dis:

1. Vanishing Gradient (VG)
2. Computationally Expensive
3. ~~No back propagation~~.

6. Leaky ReLU: (-∝, ∝)

* Solves VG + Dying ReLU problem. by small pos. slope in the neg direction (Parameterised ReLU)

* enables back prop.
* value is ∈ max(0.01x, x)

Dis:

1. Results inconsistent. for neg input values.

7. Parameterised ReLU
8. Exponential ReLU.

Softmax : Combination of multiple sigmoids.

→ sigmoid আলাদা আলাদা output এ use করব – it always gives 0 থেকে 1 – এর মধ্যে probability of a data point belonging to a class ——— but এমন না হয় সব গুলো output যোগ করলে 1 summation পাব।

But probability সব যোগ করে তো 1 পাওয়ার কথা।

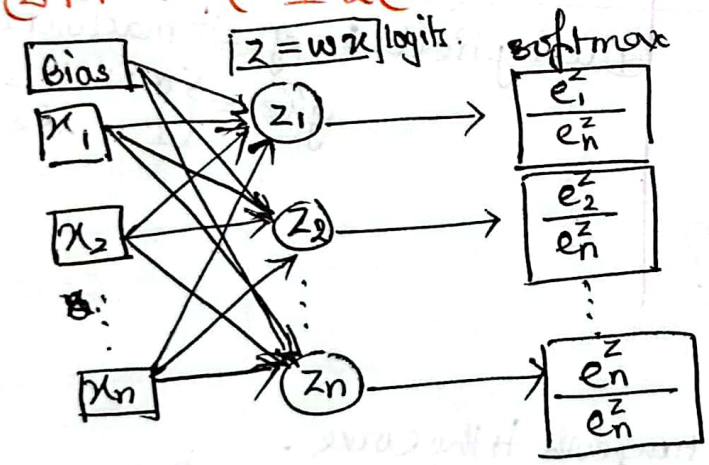
Thus sigmoid binary classification – এ use হলে 3 multiclass classification – softmax use করি।

Softmax equation → $a_i = \frac{e^{z_i}}{\sum_{k=1}^{c} e^{z_k}}$

i → individual
s → sum of output
c → no. of class.

Suppose 3 neurons gave 3 output such that 1.2, 0.9, 0.75 – but summation is more than 1

Softmax – এটা probability দিবে → 0.42, 0.31, 0.27 যোগ করলে 1 হবে।



* ** e আছে যেটা একটা monotonous function. small value কে আরও small, large value কে আরও large করতে পারে

Which activation fn is preferred?

- For CLASSIFIERS, (specially Binary) → sigmoid, tanh (but vg problem আছে)
- For Deep + high computational loaded NN → ReLU (speedy).
  (mostly used for INTERMEDIATE LAYERS)
- To avoid dying ReLU → use leaky ReLU
- Start with ReLU – then change into others.
- Softmax is used in (OUTPUT layers)

Gaviflux

* Use of softmax :-
  Distinguish blurry and crisp image.
* all layer of NN collapse into 1 linear layer if linear act. fn is used in each layer. - as it gives linear output
* HL necessity :- to learn pattern of input and feature
* Why more node in HL is better
- to learn better and give better output

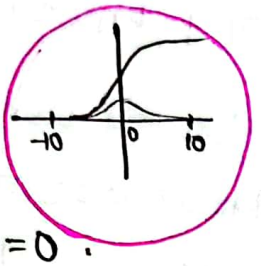
Derivatives :-

① Sigmoid :-

$$g(z) = \frac{1}{1+e^{-z}}$$

$g'(z) = \frac{d}{dz} g(z) \rightarrow$ slope of $g(z)$ at z

$g'(z) = g(z)(1-g(z))$



suppose $z = 10$.

$g(z) = 1$

$g'(z) = 1(1-1) = 0$.

or, $z = -10$.

$g(z) = 0$.

$g'(z) = 0.(1-0) = 0$

or, $z = 0$.

$g(z) = \frac{1}{2}$

$g'(z) = \frac{1}{2}(1-\frac{1}{2}) = \frac{1}{4}$.

② Tanh :-

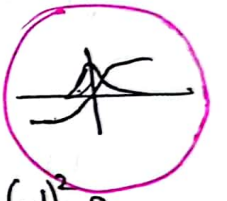
$$g(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

$$\frac{d}{dz} g(z) = \frac{(e^z+e^{-z})(e^z+e^{-z}) - (e^z-e^{-z})(e^z-e^{-z})}{(e^z+e^{-z})^2}$$

$$= \frac{(e^z+e^{-z})^2 - (e^z-e^{-z})^2}{(e^z+e^{-z})^2}$$

$$= \frac{(e^z+e^{-z})^2}{(e^z+e^{-z})^2} - \frac{(e^z-e^{-z})^2}{(e^z+e^{-z})^2}$$

$$= 1 - \tanh(z)^2$$



$z = 10$; $\tanh(z) = 1$; $g'(z) = 1-1^2 = 0$

$z = -10$; $\tanh(-10) = -1$; $g'(z) = 1-(-1)^2 = 0$

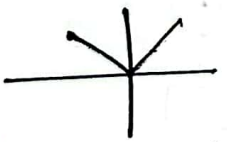
$z = 0$; $\tanh(0) = 0$; $g'(z) = 1-0^2 = 1$

③ ReLU :- $g(z) = \max(0, z)$.

$$g'(z) = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{if } z < 0 \\ \text{undefined} & \text{if } z = 0 \end{cases}$$

When derivatives doesn't exist ?

① discontinous



② more than 1 tangents to the curve.

at $x = 0$ — more than one tangent

④ Leaky ReLU :- $g(z) = \max(0.01z, z)$

$$g'(z) = \begin{cases} 0.01 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$$

Derivative exists :-

① continuous curve

② Only one tangent to the curve.

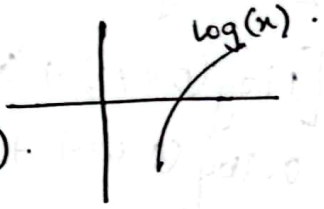
Entropy :- Indiscipline – the uncertainity inherent in the a random variable x's possible outcome.

$$H(x) = \begin{cases} -\int_x P(x) \log P(x) \cdot, & x \text{ is continous.} \\ -\sum_x p(x) \log p(x), & x \text{ is discrete} \end{cases}$$

neg sign why?
(0,1) – এর মধ্যে prob থাকে
so – log of (0,1) – এর মধ্যে যেকোনো value – ve neg value আসে – so neg-neg pos. করতে neg value থাকে

$p(x)$ গুন করতেছি cause logarithm করলে value ছোট আসে

→ $\log(p(x)) < 0$ for all $p(x)$ in $(0,1)$.



Entropy vs Information:

- Entropy: provides a measure of avg amt of info needed to represent an event drawn from a probability dist.
- Information: a way to quantify the amnt of surprise for an event measured in bits.

Example: (ডাইস)

▷=26 ○=4	▷=19 ○=16	▷=1 ○=29
①	②	③
uncertainty কম	uncertainty high – entropy high	uncertainty কম

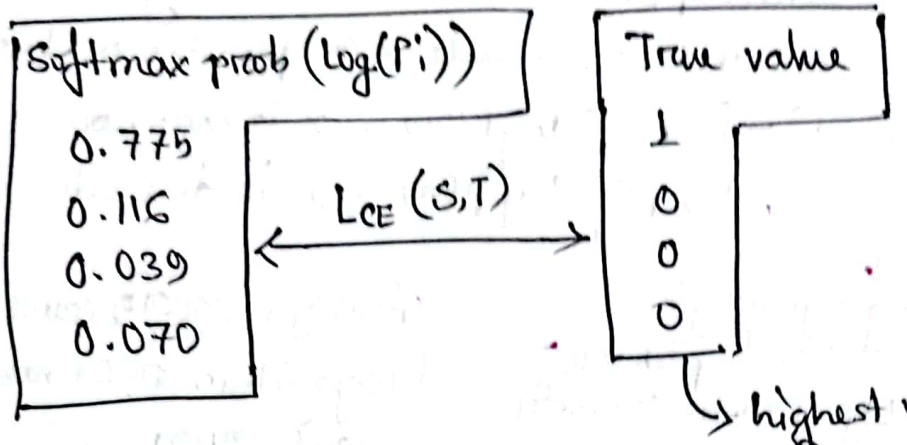
Cross Entropy / Logarithm loss fn / log loss / logistic loss fn.

$$L_{CE} = -\sum_{i=1}^{n} t_i \log(P_i) \text{ ; for n class.}$$

t_i → true, P_i → Predicted.

$$-\sum t_i \log(P_i$$

Example:



So, $L_{CE} = -\sum T_i \log(S_i)$

$= -[1\log_2(0.775) + 0\log_2(0.116) + 0.\log_2(0.039) + 0.\log_2(0.070)]$

$= -\log_2(0.775) = 0.3677$.

For ~~Binary~~ Multiclass Cross Entropy:

$$L = -\sum_{i=1}^{2} t_i \log(P_i)$$

$$= -[t\log(P) + (1-t)\log(1-P)]$$

* Catergorical Cross Entropy → one hot encoded.
* Sparse " " → integer encoded.

Viterbi Algorithm

$\pi = [\ x \quad y\ ]$

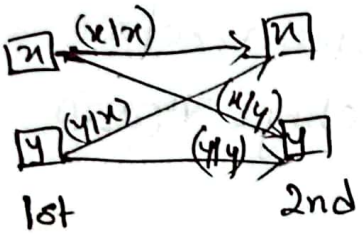
① sequence দিচ্ছে
② কোথায় state যাবে
③ for each situation x, y-এর probability বের করবে

transition matrix →

(from) \ (to)	x	y
x		
y		

Observation/ emission probability

	a	b	c	d
x	.	.	.	.
y	.	.	.	.



3rd . . . . . .

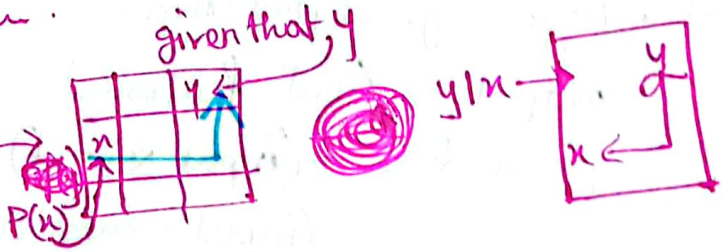
For 1st,

$P(1st, x) = P(1st|x)P(x)$
$P(1st, y) = P(1st|y)P(y)$

π. → Obsn.

For 2nd,

$P(2nd, x) = P(2nd|x)P(x|x)$
$P(2nd, x) = P(2nd|x)P(y|x)$



LR → find dw_1, dw_2, db . . .

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial a} \cdot \frac{\partial a}{\partial z} \cdot \frac{\partial z}{\partial w}$$

$$\frac{\partial L(a,y)}{\partial a} = \frac{\partial}{\partial a}\left(-(y\log a + (1-y)\log(1-a))\right)$$

$$\frac{\partial a}{\partial z} = \frac{\partial}{\partial z}(\text{activation fn}).$$

$$\frac{\partial z}{\partial w} = \frac{\partial(wx+b)}{\partial w}, \quad \frac{\partial(wx+b)}{\partial b}.$$

Find $dw_1 = \frac{\partial L}{\partial w}$ ~~> যা আসবে তা .

Gaviflux

CNN Architechecture :

* CNN output shape : $\left\lfloor \frac{\text{input shape} - \text{kernel} + 2 \times \text{padding}}{\text{stride}} \right\rfloor + 1$, out_features.

parameter : $((\text{kernel } H \times W) + 1) \times$ out_features .

* Maxpool : $\left\lfloor \frac{\text{input shape} - \text{kernel}}{\text{stride}} \right\rfloor + 1$.

parameter : 0. (no weights involved)

* Minpool : $\left\lfloor \frac{\text{input shape} - \text{kernel}}{\text{stride}} \right\rfloor + 1$

parameter: 0

* Dropout : (drops network) output → same as before

parameters => 0 , (use করে না, change করে না)

* Flatten : (multiD → 1D conversion)

output → (H × W × out_features) [shape-এর product],

parameter → 0 .

* Linear : logits / final prediction.

output → (out_features)

parameters → (input × out) if Bias = F

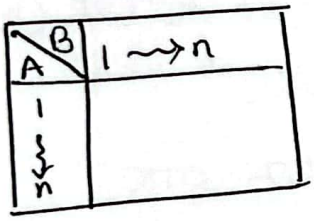
(input × out) + out if Bias = T

* Act. fn : output → out_features – একরকম

parameter → 0

	output	parameter
CNN	$\left\lfloor \frac{input - kernel + (2 \times padd)}{stride} \right\rfloor + 1$, out-f	((kernel H×W)+1)×out feat.
max/min pool	$\left\lfloor \frac{in - kernel}{stride} \right\rfloor + 1$	0
Dropout	(output-shape)	0
Flatten	(shape product)	0
Activation fn	(output shape)	0
Linear	(output shape)	(in × out) + out (B=T) (in × out) (B=F)

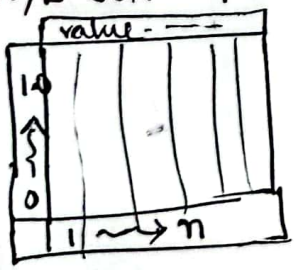
Fuzzy Extension



① সব box-এ min {A, B} বসাব.
② For n, (n-1) diagonally check করে max value নিয়ে C বানাব।

Fuzzy Addition (α cut)

① A/B box বানাব



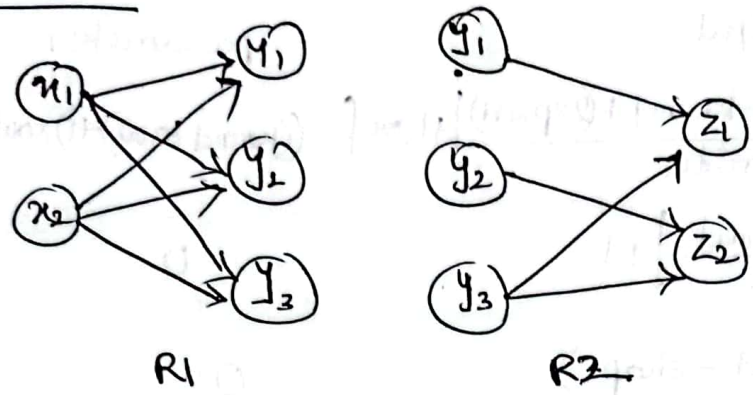
~~$C_{out} = (A_{out}$~~

② $C_{\alpha 0.1} = [A_{0.1}$ মধ্যে + $B_{0.1}$ মধ্যে, $A_{0.1}$ বামে + $B_{0.1}$ ডানে]

③ C_α[lowest, highest] – value wise box বানায় combine value বসাব (duplicate allowed, except 1).

Gaviflux

Fuzzy Relation:



x–z → R3

$\mu_{R1}(x,y)$ · (op) $\mu_{R2}(y,z)$.

Max-Avg : $\mu_{R1R2}(x_1,z_1) = \dfrac{((x_1y_1)+(y_1z_1)) \vee ((x_1y_2)+(y_2z_1)) \vee \cdots}{2}$

(N) Max-Min : $\mu_{R1R2}(x_1,z_1) = ((x_1y_1) \wedge (y_1z_1)) \vee ((x_1y_2) \wedge (y_2z_1)) \vee \cdots$

(N) Min-Max : $\mu_{R1R2}(x_1,z_1) = ((x_1y_1) \vee (y_1z_1)) \wedge ((x \text{ ——— }) \wedge - \text{ ——}$

Max-prod : $\mu_{R1R2}(x_1,z_1) = ((x_1,y_1) \times (y_1z_1)) \vee ((- \text{ ——— }) \vee - \text{ ——}$

Max-Avg → $\dfrac{(+) \vee (+) \vee (+) \cdots}{2}$

Max Min → $(\wedge) \vee (\wedge) \vee (\wedge) \cdots$

Min Max → $(\vee) \wedge (\vee) \wedge (\vee) \cdots$

Max prod → $(\times) \vee (\times) \vee (\times) \cdots$

Fuzzy membership function:

pos (/) → $\dfrac{x-a}{b-a}$; $a \le x \le b$

neg (\) → $\dfrac{a-x}{a-b}$; $a \le x < b$

cons (—) → 1 ; $a \le x < b$

(zero)

Analogy reasoning

Suppose; given, Man → Woman then king → ?

features	Women	Man	king	Queen	Apple
f1					
f2					
f3					

① Analogy = $(f_{women} - f_{man}) + f_{king}$

② let, $f_{Random} = f_{Queen}$.

③ Compare → (Random, Analogy)

Euclidean distance বের করব fR আর Analogy-র

④ এভাবে $f_R = f_{man} = f_{woman} = f_{man}$ ---, যা যার table-এ

সবকে সবার distance বের করব।

যার বলা হে king-এর কাছে থাকবে।

TF-IDF

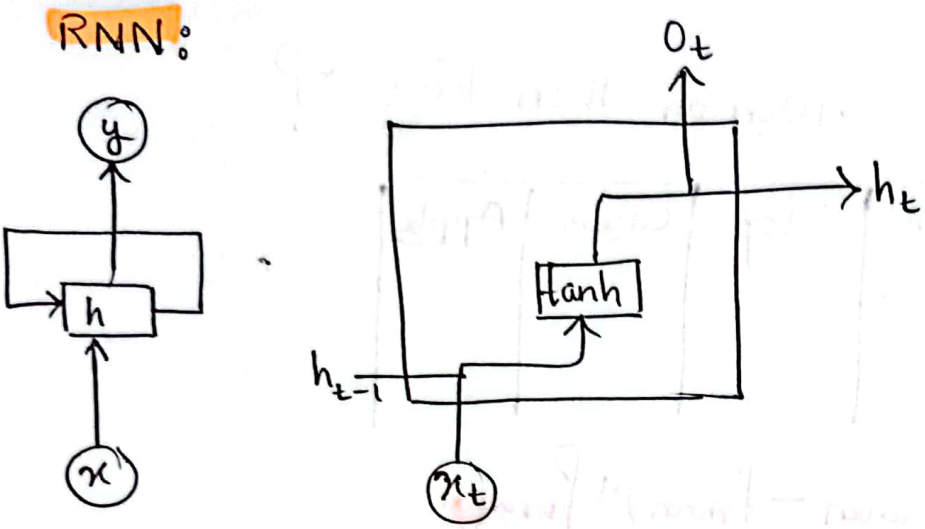
TF (Term fq) = $\frac{\text{no. of times this term appear in the sentence (Doc)}}{\text{no. of total terms in the sentence.}}$

IDF (Inverse Doc fq) = $\frac{\text{no. of total corpus}}{\text{no. of sentence this word is in}}$

term	Doc	TF	IDF	TF-IDF (product)

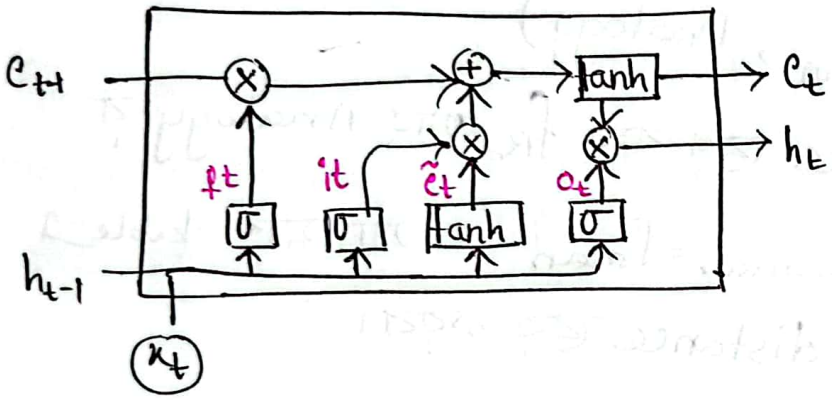
Gaviflux

RNN:



$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t)$$

LSTM:



Gates.

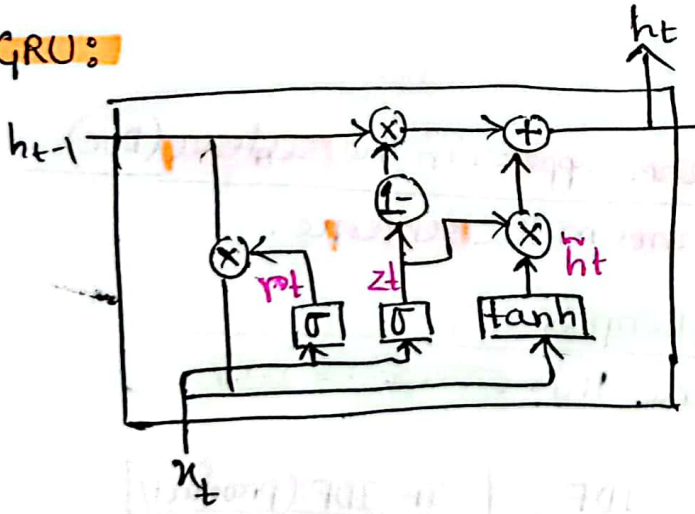
forget, $f_t = \sigma(W_f[c_{t-1}, h_{t-1}, x_t] + b_f)$

input $i_t = \sigma(W_i[h_{t-1}, x_t, c_{t-1}] + b_i)$

Candidate mem $\tilde{c}_t = \tanh(W_c[h_{t-1}, c_{t-1}, x_t] + b_c)$

Output $O_t = \sigma(W_o[c_t, h_{t-1}, x_t] + b_o)$

GRU:



(forget + input) → update gate

Gates

Update $z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$

reset $r_t = \sigma(W_r[h_{t-1}, x_t])$

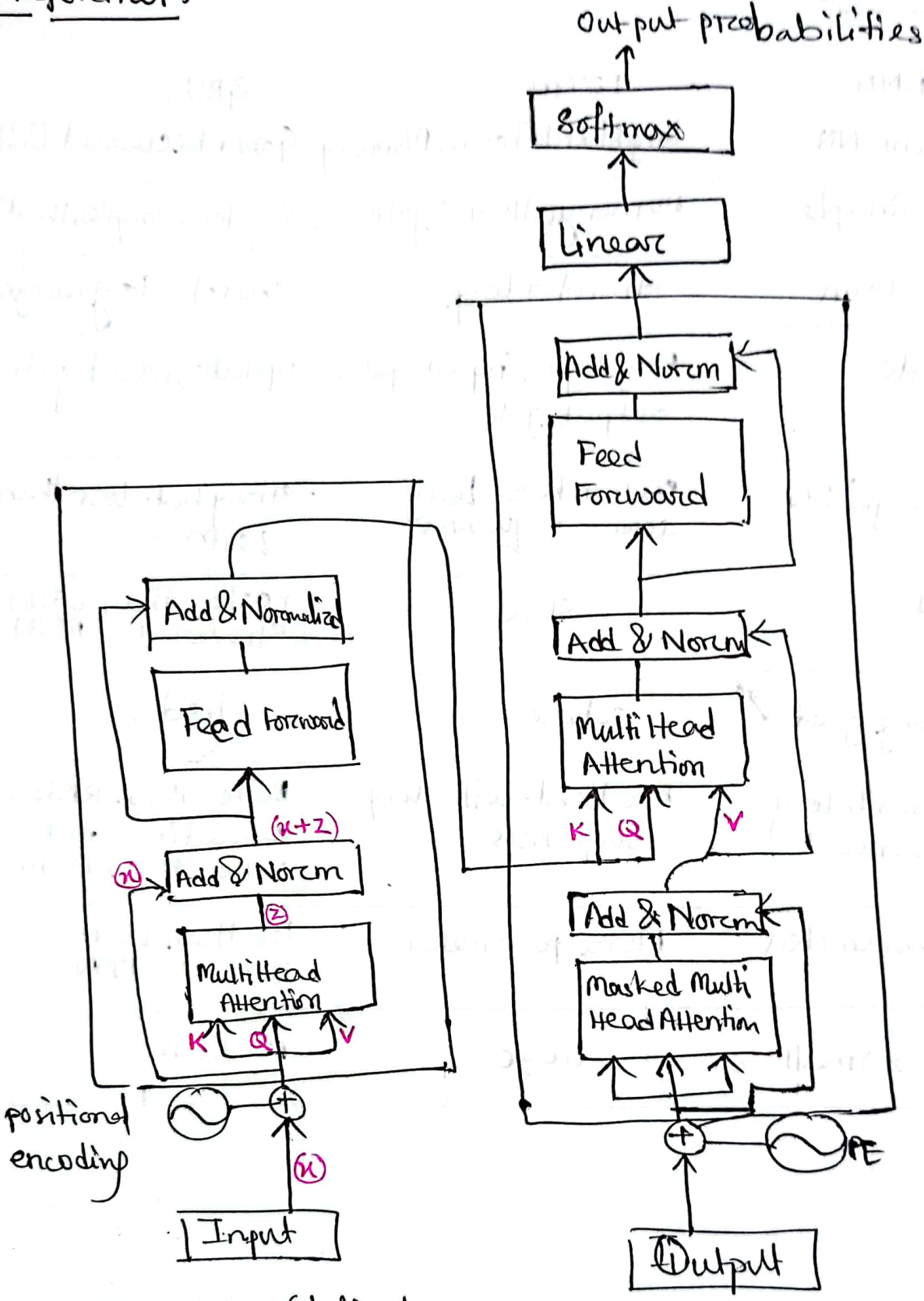
$$\tilde{h}_t = \tanh(W \cdot [r * h_{t-1}, x_t])$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Comparison:

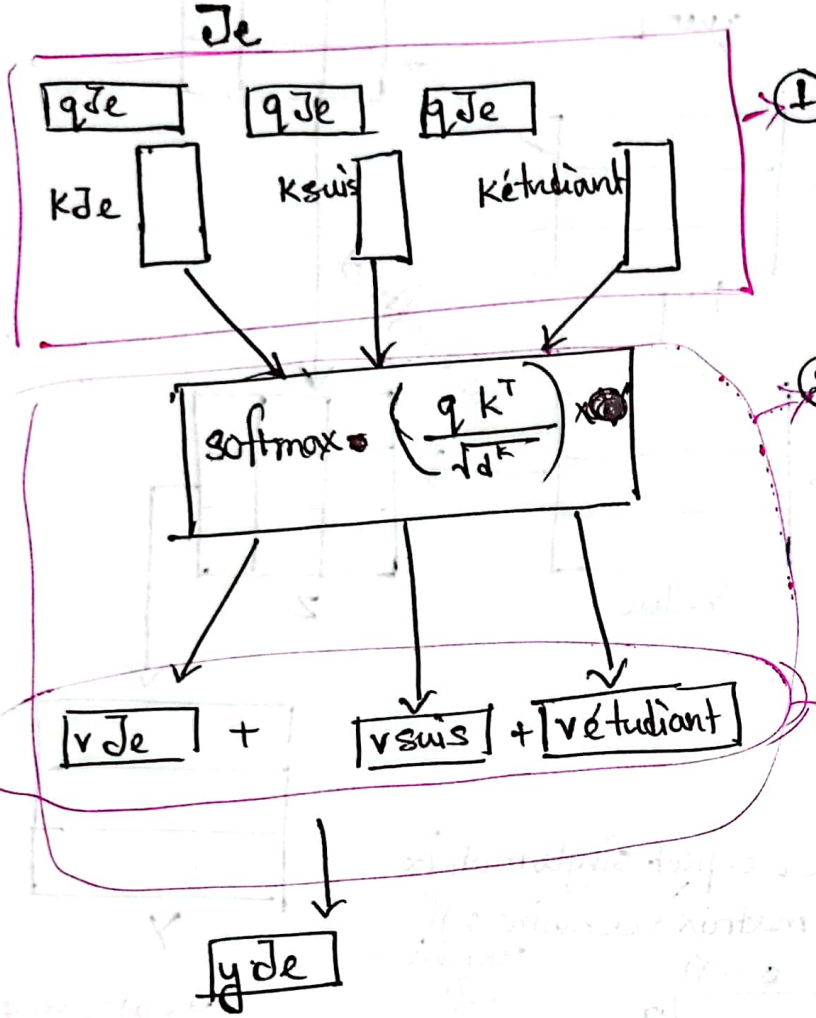
RNN	LSTM	GRU.
Recurrent NN	Long Short-Term Memory	Gated Recurrent Unit
Basic, Sinople	memorycells + 3gates	2 gates, sinpler than LSTM
short term	short + long	short - long merged
No gate	Forget gate, input gate, output gate	update, reset gate
Forgets quickly	remembers long across sequences	strong but less than LSTM
Fast	Slow	Faster than LSTM Slower " RNN
Vanishing grad ✓	solved	solved.
Poor with long sequences	Excellent with long sequences	Better than RNN, sometime not better than LSTM
Few parameter	More parameter	less than LSTM more " RNN
Model → small	large	Medium

Transformer :



Concept of Self Attention: ~~Input~~

There are three vectors of each word → Query, key, value:



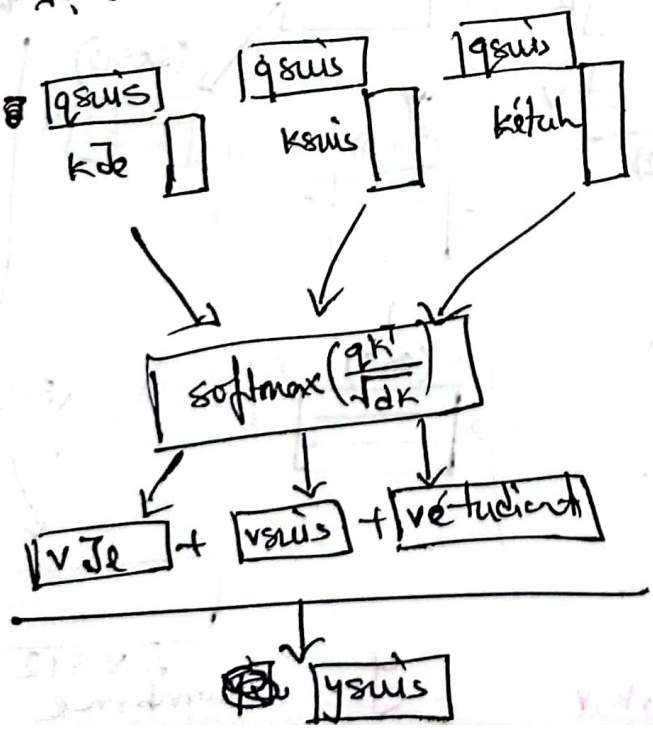
Suppose, Je

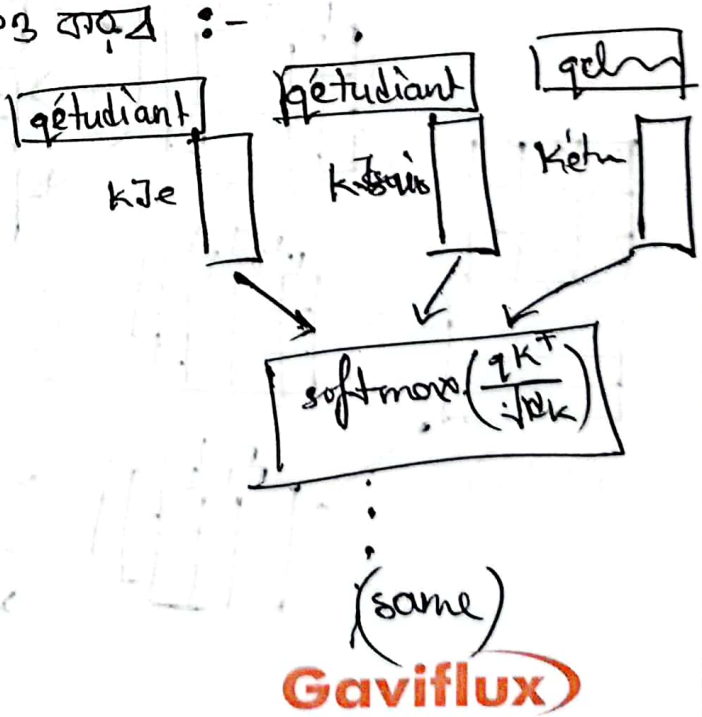
① query vector of Je will get the dot product with each key vectors of three word "Je", "suis", "étudiant"

② will ~~get~~ apply softmax on query and key vector's dot product then multiply with each value vectors to get z. ~~and then~~

③ Sum up embedding with z and proceed to feed forward.

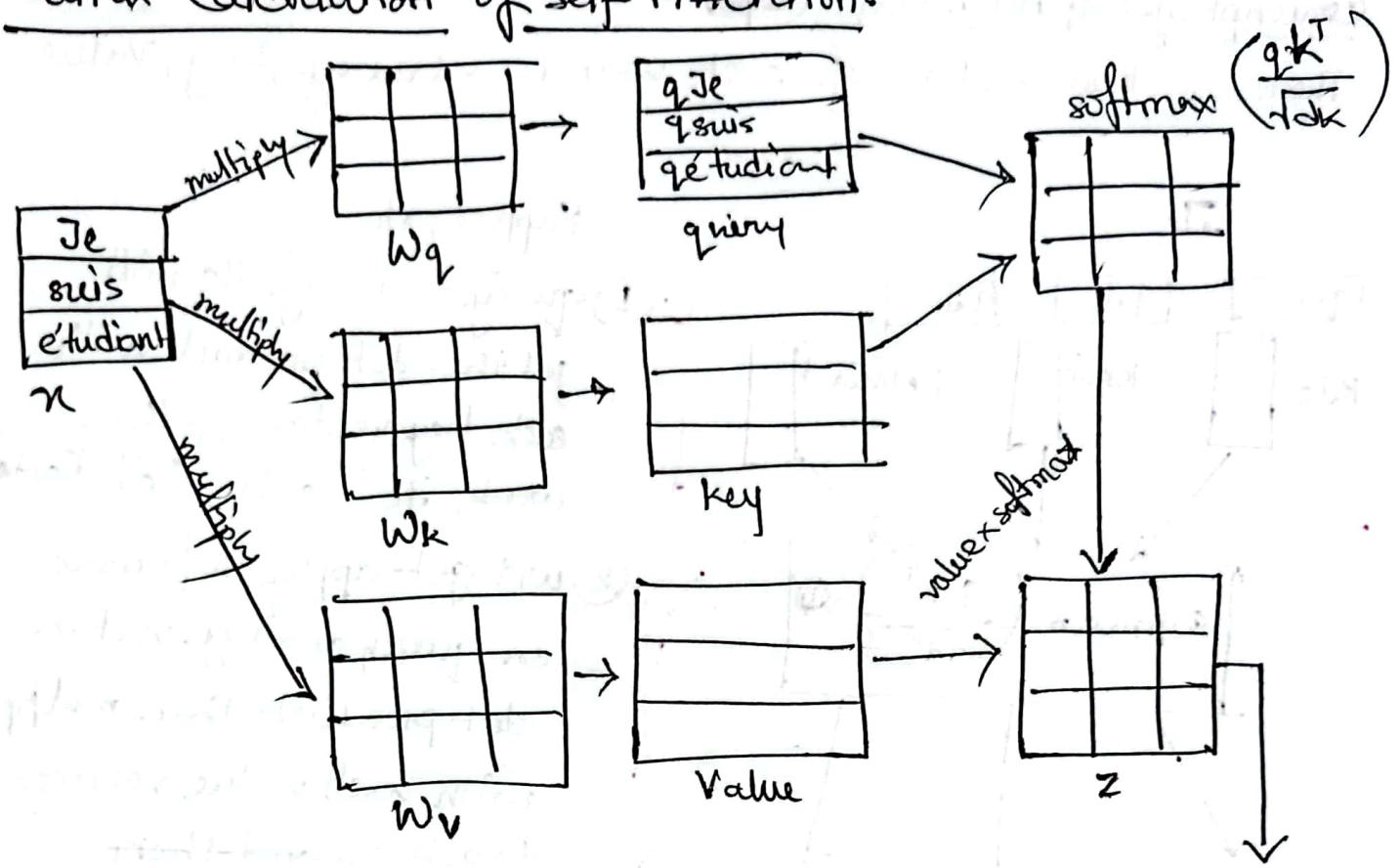
Je-র মতো suis আর étudiant -কে3 বসাই :-





Gaviflux

Matrix Calculation of self Attention:



Multihead Attention: FF layer expect single matrix not 3x3 matrix, combine self attention