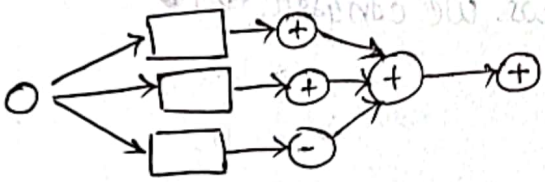
Final

Lecture-13

Ensemble Learning: process of combining supervised & unsupervised classification techniques.

→ class-imbalance data → classification accuracy.

input → classifier → individual outputs → majority voting → result/output



Ensemble classification:

$D = \{D_1, D_2, \ldots, D_N\}$

1st approach:

① Based on Rows/Instances,

① $D = D_1, D_2, \ldots, D_N$ } almost unequal size. — take a sample of equal size.

x_1-x_{35}, x_{36}-x_{76}, x_{76}-x_{100}

② Selection with/without Replacement,

with replacement 

→ we randomly select one replacement.

D1 থেকে replace হচ্ছে

D তে back যাবে then again x_2 D2 তে যাবে।

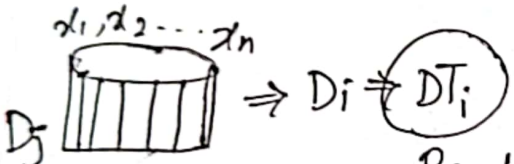
2nd Method : Based on columns / Features :

$D = \{D_1, D_2, \cdots, D_n\}$

⇒ Random subset selection

Random Forest :

consider columns/features not rows. we consider random features.



Random Tree.

$M^* = \{DT_1, DT_2, \cdots DT_n\}$

hi dimentional data

select subset of features.

Bagging:

majority voting considered.

Boosting:

weighted majority voting considered.

Input:
training data → D
# of iterations → k
learning scheme model → M.

Output: Ensemble model → M*

Method: ① loop k times:
② Di ← Dj.
③ Mi ← Dij
④ End of loop

∴ $M^* = \{M_1, M_2, \ldots M_n\}$

* Bagging algorithm, describe.

Boosting (AdaBoost):

Input: D, k, M

Output: M*

Method:

① Initialize weight of each $x \in D = 1$ or $\frac{1}{d}$ or $\frac{1}{N}$

বড় হলে 1 use করা better. ছোটকে $\frac{1}{d}$. weight of instance

② ~~for each~~

for i = 1 to k;

③ Di ← Dj; based on maximum weight of the instances

④ Mi ← Dj

⑤ error(Mi)

⑥ If error(Mi) ≥ 0.5,

⑦ go back to step 3

⑧ end if;

Error > Accuracy consider করবো না

Lecture-14

② Random forest → deals with column.

Bagging ⟶ deals with instances/rows.

Boosting / AdaBoost:

Part-1

Adaptive boosting

Input: D, K, M.

Output: M^* - ensemble

Method: ① $x_i \in D$, $w_i = 1$
↳ initialize weight

② For i=1 to k:

③ $D_i \leftarrow D$ with max. instances' weight and ~~repl~~ replacement;

D	
x_1	$w_1 = 1$
x_2	$w_2 = 1$
x_3	$w_3 = 1$
x_4	$w_4 = 1$
x_5	$w_5 = 1$.

④ $M_i \leftarrow D_i$ — building a model, M_i, from D_i, a subdataset created.

(accuracy বেশি error কম সেই model নিবো)

⑤ Compute Error of M_i on D.

⑥ If error of M_i is greater than 0.5
↓
$Error(M_i) \geq 0.5$

⑦ Goback to step 3 as the error is higher than accuracy

⑧ End if;

(9) for each correctly classified $x_i \in D$:

(10) multiply the weights of x_i by $\frac{\text{Error}}{\text{Accuracy}}$

(11) End for;

(12) Normalized the weight of every instances;

(13) End for;

(14) $M^* = \{m_1, m_2, \ldots, m_k\}$

Normalization?

↳ mathematical formula which is equal to sum of old weights ~~divided~~ divided by sum of new weights.

$$= \frac{\text{Sum of old weights}}{\text{Sum of new weights}}$$

→ misclassified এর value বেড়ে যাবে,

correctly classified → x_1, x_3, x_5 , misclassified → x_2, x_4.

Error of $m_i = \frac{2}{5} = 0.4$.

weight বের করে then normalization করবো।

{2nd iteration – 3 class
{3rd iteration → 4 class

misclassify হলে iteration continue করবো।

Part-02

① Initialize weight

$C_{yes}, W_{yes} = 0;\ C_{NO} = 0$

~~② For i = 1 to k:~~ ② for i = 1 to k:

$w_i = \log \frac{(Accuracy)}{(Error)}$

③ $w_i = \log \left(\frac{Accuracy}{Error}\right)$

④ $C_l = m_i (X_{new})$

L level বলবে Yes/No

⑤ Add w_i with C_l;

↳ Yes হলে w_i yes এ add হবে,
no হলে w_i No এ add হবে;

⑥ End for;

⑦ Return weight with largest ~~weight~~ weight.

(*) result যা হবে, সেই C level এ add হবে।
weighted majority voting করে হবে,

yes হলে, $C_1 = 0 + 1 + 1$ (w_1 w_3)

no হলে $C_1 = 0 + 1$ (w_2)

Weighted majority voting দেখবো।

Lecture-15

Python: super-high level programming language.

- python.org → must explore official site.
- jupyter.org → Jupyter notebook
- Anaconda → package contains jupyter, er and many other softwares.
- google.colab → platform providing online jupyter notebook
  - ↳ provides GPU service.

Basic Python:

Data types in Python?

mainly 5 types of datatypes in Python.

1. Boolean → False, True
2. Integer → all real numbers
3. Float → all float (no double) (fraction)
4. String → '', "", character & string same.
5. Complex → (e.g. 2+3j)

Java
- 2 datatypes.
  - Primitive (8)
  - Ref (2)

total 10 type data

Data Structure in Python? Why we need?

1. List / Array (consider in python)
2. set
3. Tuple
4. Dictionary

documentation comment = ###

Lecture-16

Python

Data Types	Data Structure
① Boolean	① List
② Integer	② Tuple
③ Float	③ set
④ string	④ Dictionary
⑤ complex	

```
var = False  → Bool
var = 'UIU'  → str
var = 10     → int
```

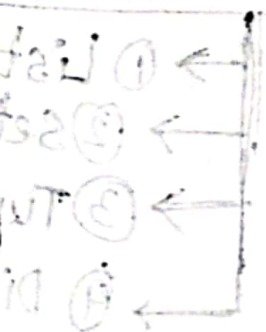
```
my_list = [ ]
print(my_list)
```

```
my_list = ['UIU', True, 2]
my_list = [2, 6, 8, 10]
type(my_list)
↳ list
```

```
my_tuple = (2,3,6)
my_set = {2,2,3,2,4}
print(my_set)
↳ output: 2,3,4.
```

nested dictionary :

```
my_dict = {1: {1:'cat', 2:'rat'},
           2:'Java',
           3:'Python'}
```



Pandas

Exploring dataset with panda in Python (EDA / (Exploratory Data Analysis)

Lecture

NumPY — H/W.

EDA — matplotlib / seaborn...

Logistic Regression
sklearn

Deep learning (chp-3)

↳ subset of machine learning

↳ basically artificial neural network (ANN)

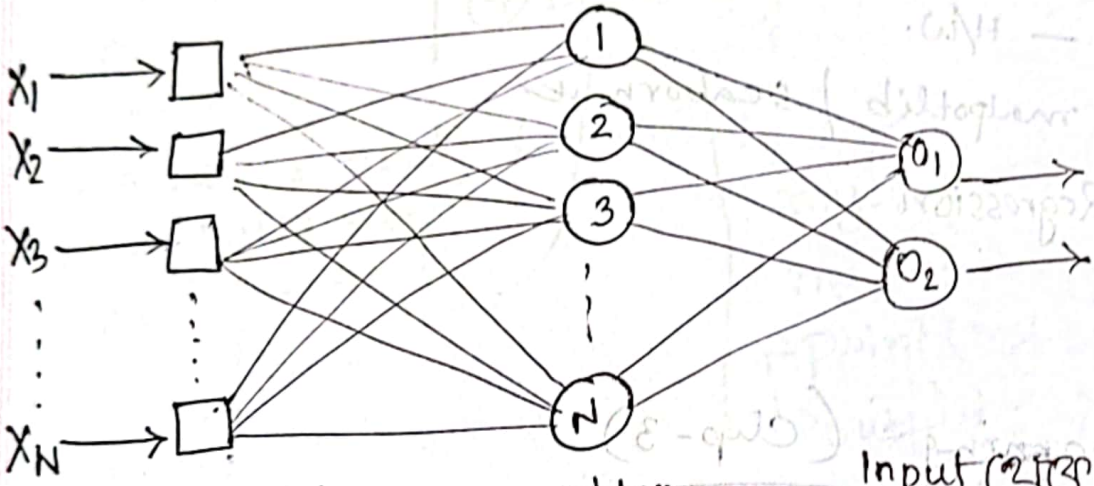
Traditional ML → defines set of features of data.

LR, classification
KNN - ...

→ মানুষের human brain এর মত কাজ করতে চায়।

3.2 Building neural networks

① Input layer
② Hidden layer
③ Output layer



↓ feature এর value input নেয়

Input layer

fully connected

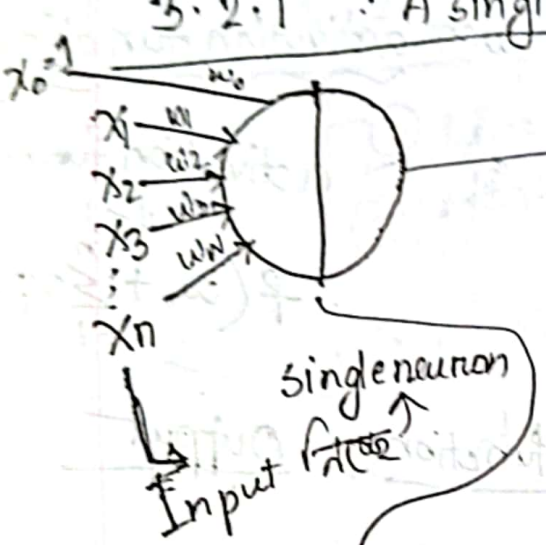
Hidden layer

Input থেকে hidden layer যা learn করেছে, তাই output এ পাঠিয়ে দিচ্ছে।

more than one hidden layer → deep learning.

one hidden layer ⟶ artificial neural network (ANN).

3.2.1 : A single Neuron



w_0 = bias.

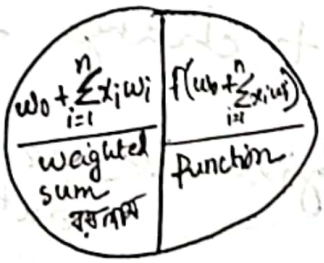
$$\hat{y} = f(w_0 + X^T W)$$

$$= f\left(w_0 + \sum_{i=1}^{n} x_i w_i\right)$$

এই single neuron এ দুটো কাজ হয়।

① weighted sum

② function.

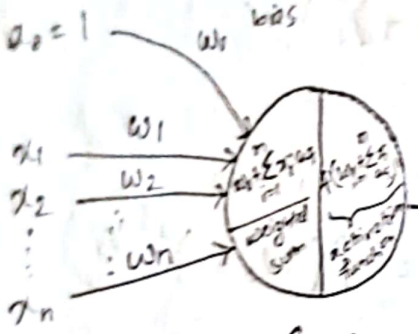


Ⓐ difference between চাওয়া-পাওয়া change weight.

Input weight Sum Activation function. Output.

lecture

Neuron Forward Propagation with activation function:



: activation function

$$\cdot f\left(w_0 + \sum_{i=1}^{n} x_i w_i\right)$$

Input · weights · Sum · Activation function · Output

forward-propagation → just needs to change the weights.
→ input given → we know what output we want.
→ adjusts the weight forward.

back-propagation → starts with output and goes to sum/activation function. goes backword.
↳ adjusts weight backward.

Activation function :

① Linear activation function.
② Non-linear function.
activation.

decides whether neuron should be activated or not by calculating weight.

In linear activation function → $\hat{y}$ = weighted sum.

↓

when input & output same & needs no transformation.

Non-linear activation function :

① Sigmoid function ; $\sigma(x)$
② Hyperbolic tangent function, $\tanh(x)$
③ Rectified Linear Unit (ReLU)

Sigmoid function: → logistic function.

$\sigma(x)$

→ formula: $\sigma(x) = \frac{1}{1+e^{x}}$; where x is weighted sum.

→ squash result between 0 and 1.

input large & positive → output → 1.
input large & negative → output → 0.

Hyperbolic Tangent:

↳ returns either -1 or 1.

↳ defined by formula;

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

ReLU: if the total weighted sum is negative, it returns 0.

less than 0 or negative → returns → 0
positive or more than 0 → return → x
equals to 0 → 0.

Forward Propagation : move from input to output.

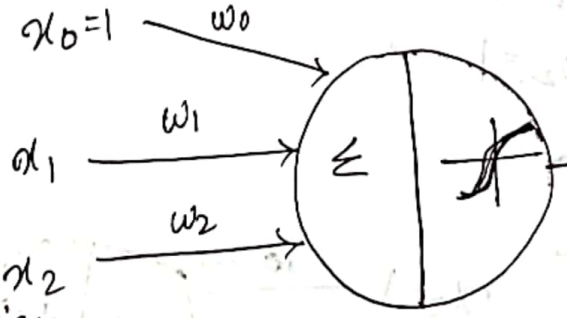
AND Gate

x_1	x_2	Y
0	0	0
0	1	0
1	0	0
1	1	1

read

— 0, 1 → binary class

sigmoid ✓



gradient

only the weights are not given.

random 0 assign.

$w_0 = 0$
$w_1 = 0$
$w_2 = 0$

$x_0 = 1$ $w_0 = 0$
$x_1 = 0$ $w_1 = 0$
$x_2 = 0$ $w_2 = 0$

0 → $\hat{y} = 1$

output positive.

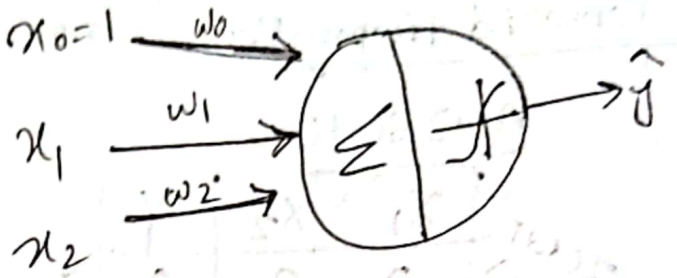
(*) learning rate, value of weight change হবে।

(*) এমন একটি weight বের করতে হবে, যেটা 4 টাকেই satisfy করবে।

$w = 0, -1, 1, 2, -2$ ইহা can be any need to satisfy all 4 conditions

OR Gate

x_1	x_2	Y
0	0	0
0	1	1
1	0	1
1	1	1



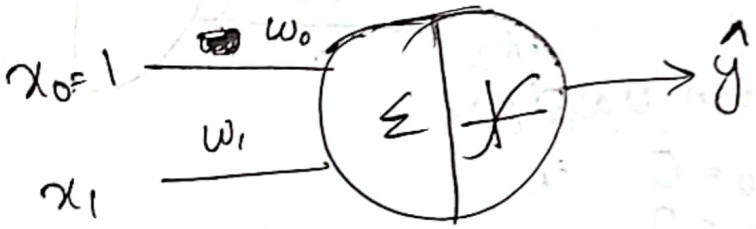
$w_0 = -1$

$w_1 = 1$

$w_2 = 1$

NOT Gate:

x_1	Y
0	1
1	0



$w_0 = 1$

$w_1 = -2$

07-09-22

Lecture

3.2.3 The Perceptron.

3.2.4: Understanding Loss Functions.

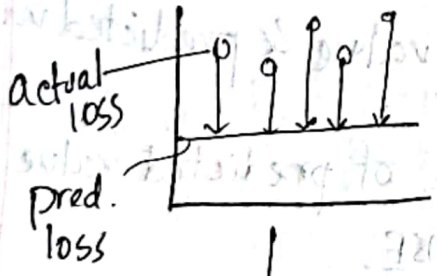
Loss function: difference between actual value and predicted value.

$L = y_i - \hat{y}_i$ → penalty for misclassification of a learning model.

Total loss $= \sum_{i=1}^{n} (y_i - \hat{y}_i)$

$L = y_i - \hat{y}_i = 0$ → correctly classified.

Loss
↓
error
↓
we need to minimize the error



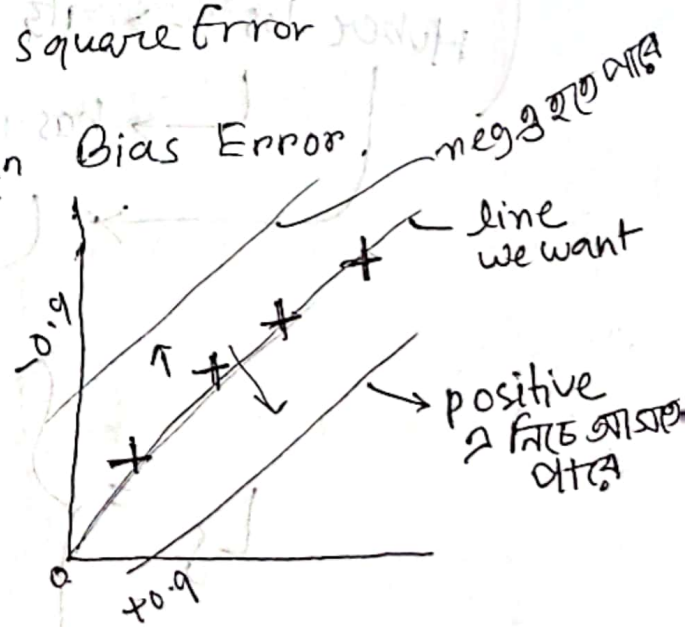
High loss.

figure-3.13.

Loss function
- Mean Absolute Error
- mean square Error
- Mean Bias Error.
- MSLE

it should be close to 0.

negative থাকলেও সমস্যা।



use for regression problem.

MAE → L1 loss → use it for regression model.

MSE → L2 loss → we are using square and taking the average.

MBE → calculate average (bias) in the model.

- → value negetive এবং positive হতে পারে
- → model may be biased towards a particular examples

Underestimating
overestimating
a parameter.

difference between actual value & predicted value.

MSLE → natural logarithm of predicted value is used. similar to MSE.

Huber los → its linear & for small error
- → has 2 values.
- → (δ) ← characterised by

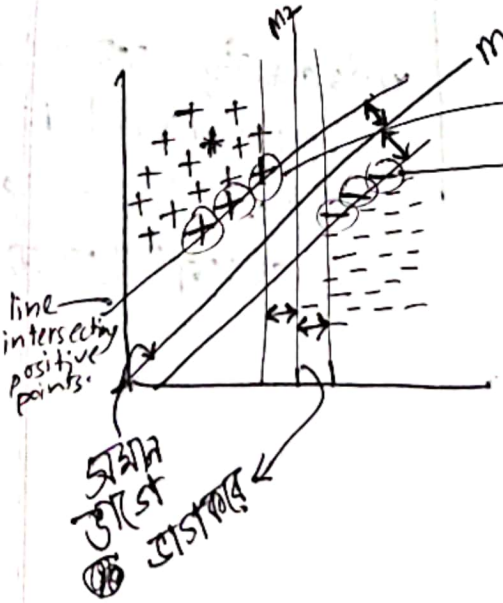
$$L_\delta = \begin{cases} \frac{1}{2}(y_i - \hat{y}_i)^2, & \text{if } (y_i - \hat{y}_i) \le \delta \\ \delta(y_i - \hat{y}_i) - \frac{1}{2}\delta^2 & \end{cases}$$

— slide

Binary Cross Entropy Loss → between 0 and 1
→ for binary class classification
→ calculates average diff. between predicted & actual probabilities.

Categorical Cross Entropy loss → multi-class classification এর জন্য।
→ প্রবাহিত zero & other element multiplied by 0.
→ softmax

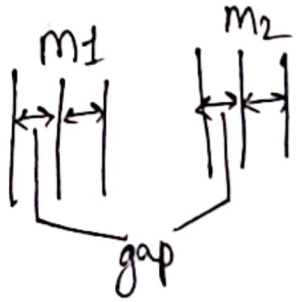
Hinge loss : → developed for SVM.
→ penalise wrong prediction.



support vector for positive examples
support vector for negetive examples.

m_1 → decision line

* যার gap যত বেশি সে তত ভালো।



P=projection

andvengine
SVM
Youtube video.
দেখে নিবা।

$$\text{SVM Loss} = \sum_{j \neq j} \max(0, s_j - s_{j,} + 1)$$

gap বাড়িয়ে দিচ্ছে, error বাড়িয়ে দিচ্ছে।

Gradient Descent (algorithm)

Except bias - all are between 0 to 1 weights given

bias কে usually বড় রাখতে চাই
0.001 --

আসতে আসতে আসবে।

classify correctly → 0
misclassify হলেই → 1

pred − actual + 1
(1 − 0) = 1 + 1
= 2
misclassify

correctly classify করলে যদি 100 টার অর্থ 0 হবে, total sum 0 চাই।

misclassify 0.5 এর বেশি হলে model নিবো না, 50% misclassify accept করবো না।

Derivatives - Basic Intro:

$$\frac{d}{dx}[c] = 0$$

- $\frac{d}{dx}[7] = 0$
- $\frac{d}{dx}[\pi] = 0$
- $\frac{d}{dx}[\pi^2] = 0$

Power Rule:

$$\frac{d}{dx}[x^n] = n \cdot x^{n-1}$$

$$\frac{d}{dx}[x^5] = 5x^4$$

$$\frac{d}{dx}[4x^5] = 4 \cdot 5x^4 = 20x^4$$

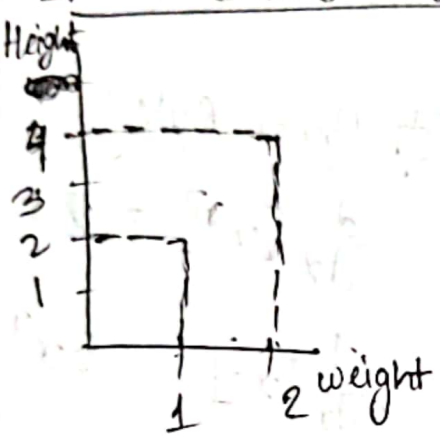
$$f(x) = 4x^3 + 7x^2 - 9x + 5$$
$$= 12x^2 + 14x - 9 + 0$$

$$\frac{d}{dx}(f(g[x])) = f'[g(x)] \cdot g'(x)$$

$g'(x)$ — derivation of inside function.

$$\frac{d}{dx}[x^2 - 3x]^5 = 5 \cdot [x^2 - 3x]^4 \cdot (2x - 3)$$

The Chine Rule:

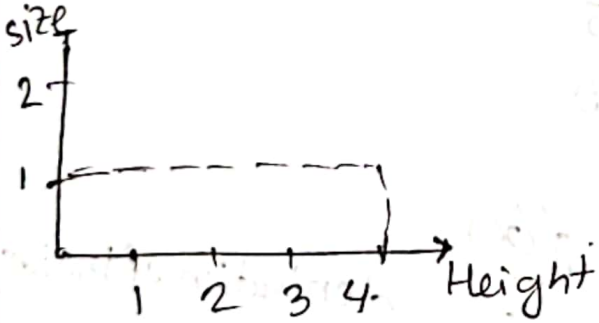


weight : Height
1 : 2.

$$\frac{d\,\text{Height}}{d\,\text{weight}} = 2.$$

The equation for height $= \frac{d\,\text{height}}{d\,\text{weight}} \cdot \text{weight}$

$= 2 \cdot \text{weight}.$



4 unit of height = 1 unit of size.

$$\frac{d\,\text{size}}{d\,\text{height}} = 1/4$$

The equation for size $= \frac{d\,\text{size}}{d\,\text{height}} \cdot \text{height}$

~~$= 1/4 \cdot \text{height}$~~

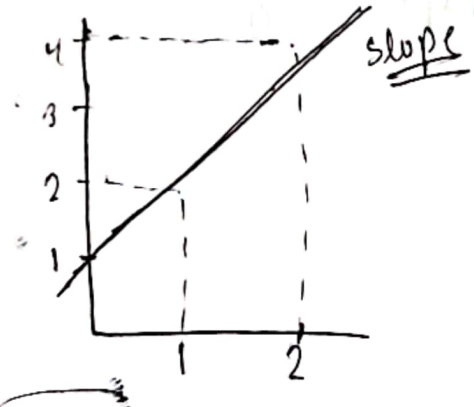
$$= \frac{d\,\text{size}}{d\,\text{height}} \times \frac{d\,\text{height}}{d\,\text{weight}} \times \text{weight}.$$

Therefore, the relation is the essence of the chain rule:

~~$= \frac{d\,\text{size}}{d\,\text{weight}}$~~

$$\frac{d\,\text{size}}{d\,\text{weight}} = \frac{d\,\text{size}}{d\,\text{height}} \times \frac{d\,\text{height}}{d\,\text{weight}} = \frac{1}{4} \cdot 2 = \frac{1}{2}.$$

That means every one unit increase of weight increases.
1/2 unit of size.



$\hat{y}$ = Intercept + (slope × x)

Height = Intercept + (slope × weight)

The chain Rule applies to the **Residual Sum of squares**

→ common loss function in machine learning.

$y = (\text{Residual})^2$

$= (\text{Observed} - \text{Predicted})^2$

Let's consider,
slope = 1.
Height = Intercept + (1 · weight)

The derivative is 0 at the lowest point.

$\frac{d\,\text{Residual}^2}{d\,\text{Intercept}}$ ~~$= \frac{d(\text{Residual})^2}{d\,\text{Residual}}$~~

$= \frac{d\,\text{Residual}^2}{d\,\text{Residual}} \times \frac{d\,\text{Residual}}{d\,\text{Intercept}}$

The Power Rule of Residual^2 is:

$\frac{d}{d\,\text{Residual}} \cdot \text{Residual}^2 = 2 \cdot \text{Residual}$

$\frac{d\,\text{Residual}^2}{d\,\text{Intercept}} = 2 \times \text{Residual} \times \frac{d\,\text{Residual}}{d\,\text{Intercept}}$.

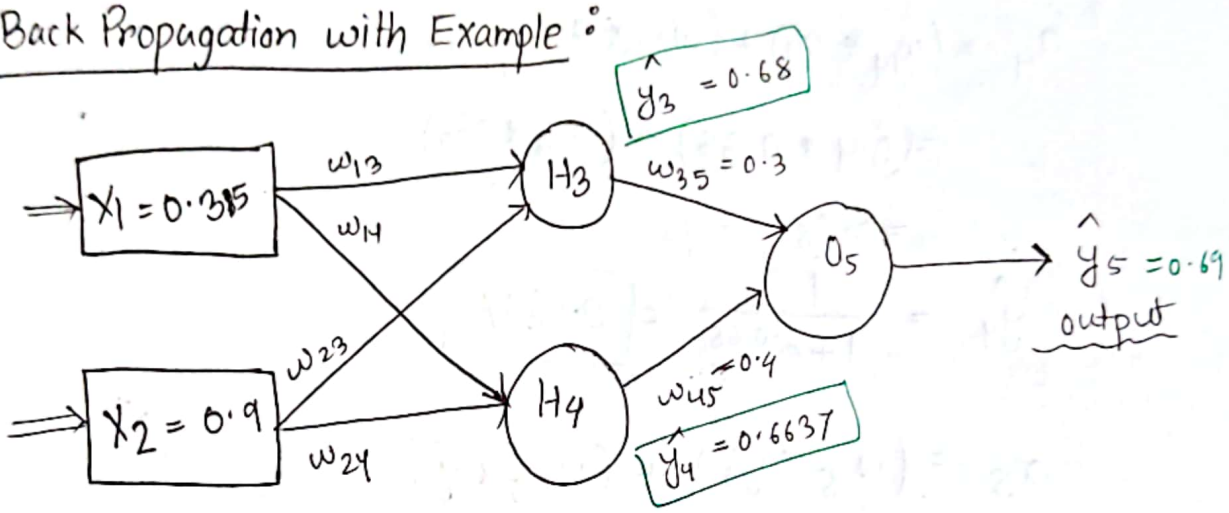
$$\text{Residual} = (\text{Observed} - \text{Predicted})^{2}$$

$$= \text{Observed} - \text{Intercept} - \text{weight}$$

$$\frac{d\,\text{Residual}}{d\,\text{Intercept}} =$$

Lecture

Back Propagation with Example :



Let's assume,

Actual output, $y = 0.5$

Activation function, Sigmoid $= \frac{1}{1+e^{-a}}$

Learning rate, $\eta = 1$

- Forward Propagation:

$$a_i = \sum_j w_{ij} \cdot a_j$$

$$y_i = f(a_j) = \frac{1}{1+e^{-a_j}}$$

So, now, $a_3 = (w_{13} * x_1) + (w_{23} * x_2)$

$= (0.1 * 0.35) + (0.8 * 0.9)$

$= 0.755$

$$\hat{y}_3 = \frac{1}{1+e^{-0.755}} = \boxed{0.68}$$

$$a_4 = (w_{14} * x_1) + (w_{24} * x_2)$$

$$= (0.4 * 0.35) + (0.6 * 0.9)$$

$$= 0.68$$

$$\hat{y}_4 = \frac{1}{1+e^{-0.68}} = \boxed{0.6637}$$

$$a_5 = (w_{35} \times \hat{y}_3) + (w_{45} \times \hat{y}_4)$$

$$= (0.3 \times 0.68) + (0.9 \times 0.6637)$$

$$= 0.801$$

$$\hat{y}_5 = \frac{1}{1+e^{-0.801}} = 0.69$$

$$\text{Error} = y - \hat{y}$$

$$= 0.5 - 0.69$$

$$= -0.19$$

Each weight changed by :

$$\Delta w_{ji} = \eta \delta_j o_{ji}$$

$$\boxed{\delta_j = o_j(1-o_j) \cdot (t_j - o_j)}$$ — if j is an output neuron

$$\boxed{\delta_j = o_j(1-o_j) \sum_k \delta_k w_{kj}}$$ — if j is hidden neuron

where, η is a learning rate

t_j is the target value or actual value.

δ_j is the error for j neuron.

For output neuron :

$$\delta_5 = O_5(1-O_5)\cdot(t-O_5)$$

$$= \hat{y}_5(1-\hat{y}_5)(y-\hat{y}_5)$$

$$= 0.69\,(1-0.69)\,(0.5-0.69)$$

$$= -0.0406.$$

For hidden neuron,

$$\delta_3 = \hat{y}_3(1-\hat{y}_3)\cdot\delta_5\cdot w_{35}$$

$$= 0.68(1-0.68) * 0.3 * (-0.0406)$$

$$= -0.00265.$$

$$\delta_4 = \hat{y}_4(1-\hat{y}_4)\cdot\delta_5\cdot w_{45}$$

$$= 0.6637(1-0.6637) * 0.9 * (-0.046)$$

$$= 0.0082.$$

$$\Delta w_{ji} = \eta\,\delta_j\,o_i.$$

$$w_{45} = \eta\cdot\delta_5\cdot y_4$$

$$= 1*(-0.0406) * 0.6637$$

$$= -0.0269$$

$$w_{45}(new) = \Delta w_{45} + w_{45}(old)$$
$$= -0.0269 + 0.9$$
$$= 0.8731$$

~~$\Delta w_{14} = \eta \delta_4$~~

$$\Delta w_{14} = \eta \delta_4 x_1$$
$$= 1 * (0.0082) * 0.35 = -0.00287.$$

$$w_{14}(new) = \Delta w_{14} + w_{14}(old)$$
$$= -0.00287 + 0.4$$
$$= 0.3971$$

~~w_{12}~~
w_{23} (H.W)
w_{24}

Similarly update all other weights :

i	j	w_{ij}	δ_j	x_i	η	Updated w_{ij}
1	3	0.1	-0.00265	0.35	1	0.0991
2	3	0.8	-0.00265	0.9	1	0.7976
1	4	0.4	-0.0082	0.35	1	0.3971
2	4	0.6	-0.0082	0.9	1	0.5926
3	5	0.3	-0.0406	0.68	1	0.2726
4	5	0.4	-0.0406	0.6637	1	0.8726

$$a_3 = (w_{13} * x_1) + (w_{13} * x_2)$$

$$= (0.0991 * 0.35) + (0.1916 * 0.9)$$

$$= 0.7528$$

$$\hat{y}_3 = \frac{1}{1+e^{-0.7528}} = 0.6797$$

$$a_4 = (w_{14} * x_1) + (w_{24} * x_2)$$

$$= (0.3911 * 0.35) + (0.5926 * 0.9)$$

$$= 0.6723$$

$$\hat{y}_4 = \frac{1}{1+e^{-0.6723}} = 0.6620$$

$$a_5 = (w_{35} \times \hat{y}_3) + (w_{45} \times \hat{y}_4)$$

$$= (0.2726 \times 0.6797) + (0.8126 \times 0.6620)$$

$$= 0.7631$$

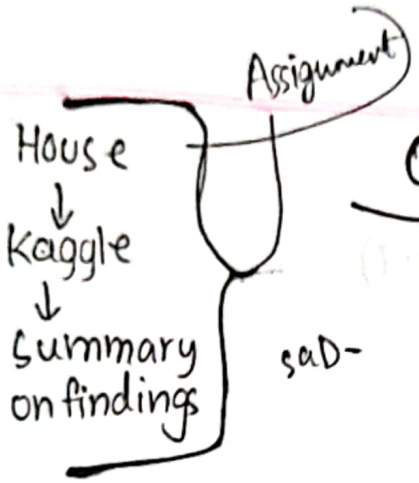
$$y_5 = \frac{1}{1+e^{-0.7631}} = \boxed{0.6820}$$

new network output

$$\text{Error} = y - \hat{y}$$

$$= 0.5 - 0.6820$$

$$= -0.182$$



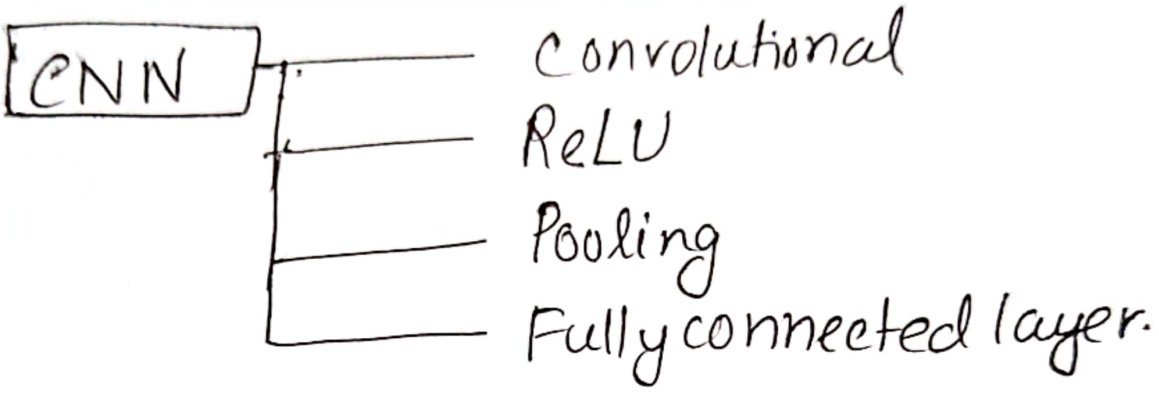
Gradient Descent Algorithm.

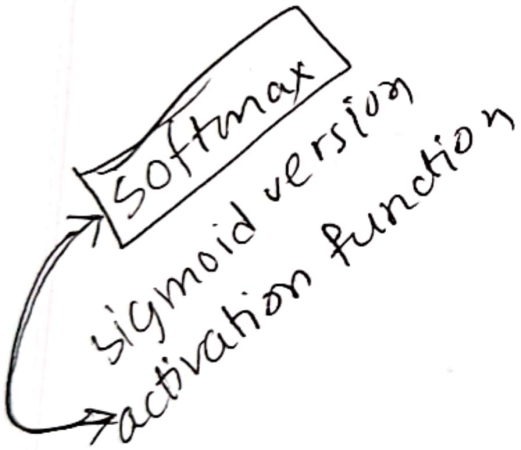
Regularization 1 : Dropout → to reduce Overfitting issue.

SNA → search Network Architecture.
↳ which nauron and which architecture to use.

Regularization Early Stopping :

- training model & minimizing loss on validation set.
- when test starts increasing we stop the training model & that is early stopping.


CNN
convolutional
ReLU
Pooling
Fully connected layer.


Softmax
sigmoid version
activation function