Informed search

Thursday, December 5, 2024 11:26 PM

Heuristic function (Rule of thumb)

it means guessing.

It is a technique to solve a problem quickly.

* many methods to calculate. One of them

→ Eucledian distance. $\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$

as it guess, so it guarenty's to give the good solution but not the optimal one.

Blind search gives the optimal solution.

So, want faster solution?

→ use this method, the greedy one.

CNN
Friday, December 6, 2024 2:08 PM

"CNN"

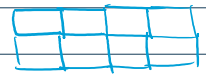
it is basically a special type of ANN, specially designed for processing structure grid like images. widely used in

→ Image classification
→ Object detection.
→ Computer vision.

Shortly if we have grid like structure like

1D

or,



2D then we can use CNN.

In CNN there are

- Convolution layer & operation
- Pooling (Max, Min, Avg)
- stride, hyperparametre,
- fully connected layer (In ANN) also.

CNN is inspired from Visual Cortex.

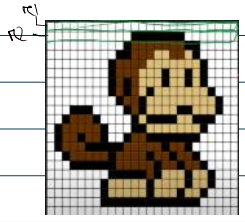
Now CNN vs ANN :→

1. Can't we use ANN on images?

Ans:- yes, we can. But it will cost us:

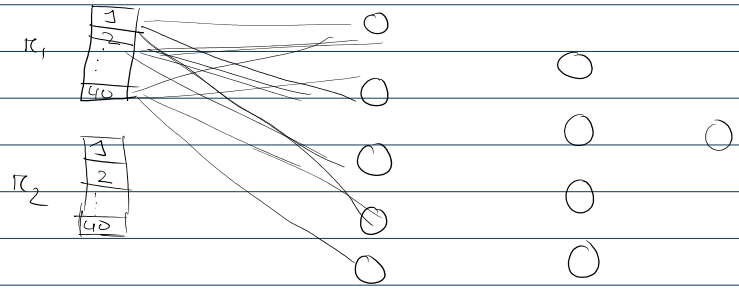
a. High computational cost
b. overfitting
c. Loss of imp info like spatial arrangements of pixels.

Still lets take an example:-



40x40

we want to use ann here. say its a 40 by 40 image, means there are 40 rows. each rows consisting 40 numbers. In order to apply ann lets take this 2D to 1D.



100 units

Comparing CNN Vs ANN | CampusX

Here, even though we have taken a 40x40 image which is very

100 units

Here, even though we have taken a 40x40 image which is very small, hidden layer of 100 units (also very small) still would need 1 lakh 60,000 weights. Now imagine for a 1000 x 1000 image?
very high computational time & cost.

b. Overfitting:

as we are connecting very minutely with each node high chance of Overfitting. Train data te valo result dileo test data te dibe na.

c. Loss:

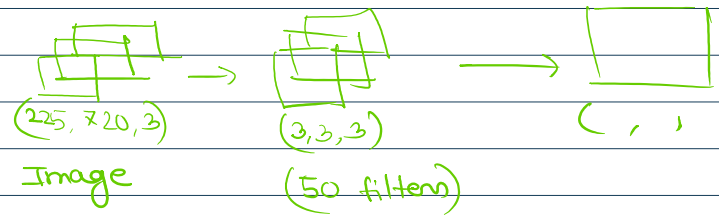
in image like the distance between eye and nose is also very imp. converting from 2D to 1D, high chance of losing this.

So, ANN is not used here.

In CNN, we use filters to separate each thing.

* One more difference:-

say,



Calculate the parametres here in CNN?

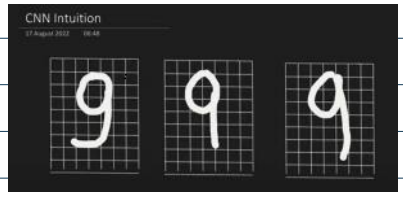
filters are learnable parametres in CNN. so, 3x3x3 = 27 weights
x 50
1350
+ 50 - bias
1400

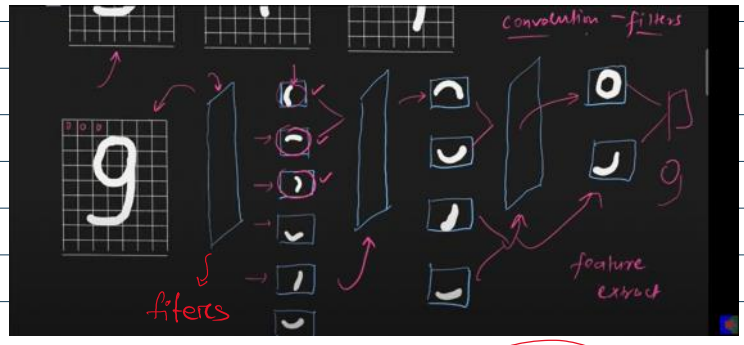
Again if we take larger image keeping the filter size same, the learnable parametres would still be same, 1400.

So, we understand, in CNN parametres are not dependent on image size.
so, computational cost here is low.

CNN Intuition:-

CNN Intuition :-





Applications :-

→ location of object in image.
→ Image detection
→ facial recognition
→ image pigmentation.

Can watch from here too :-

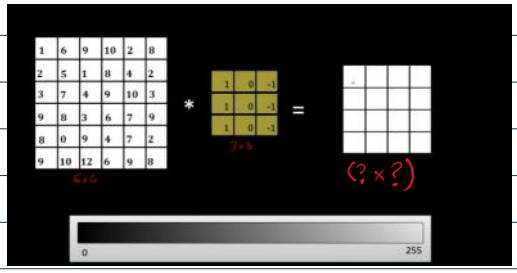
What is Convolutional Neural Network (CNN) | CNN Intution

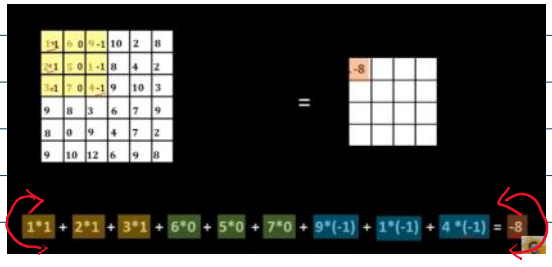
Translation → is a geometric transformation that shifts all points by same distance.

Translation variance → is a property of a system/model where its performance or output does change when the input data is translated / shifted. We need translation invarience.

Translation Equivarince → is a property of a function or an operation where the output structure does not change even if the input is translated.

"CNN operations"

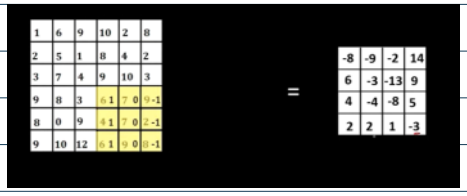




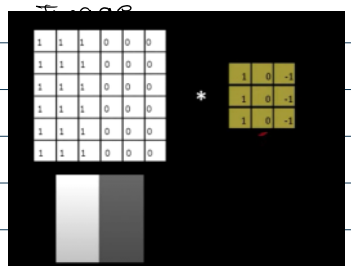
$(n \times n) * (f * f) = (n-f+1) \times (n-f+1)$

$(n \times n) * (f * f) = (n-f+1) \times (n-f+1)$

$\Rightarrow (6 \times 6) * (3 \times 3) = 4 \times 4$

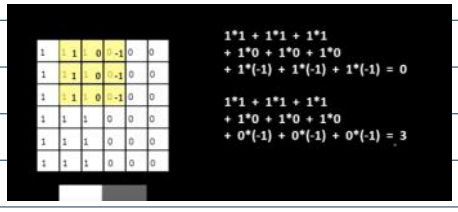


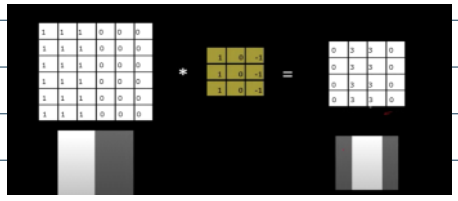
Example:-



Here, 1 ⇒ white

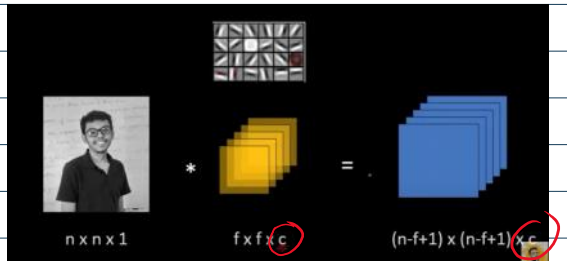
0 ⇒ black.





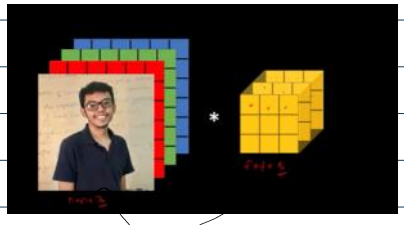
black and white image for one filter.

If we have a picture then different filters would extract different edge,



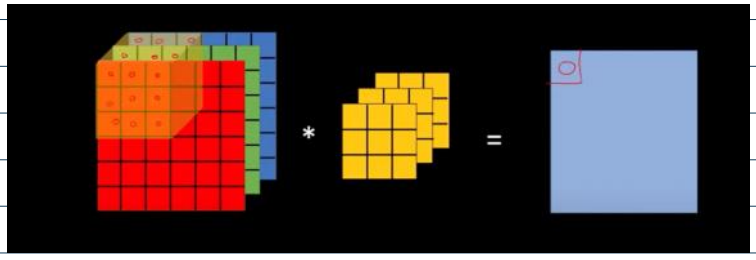
And in case of colorful picture,

rgb te 3 ta frame usally, (R, G, B).



should match

⇒



gonna multiply and add all 27 value to generate one value.

Here,

$n \times n \times 3$ $\quad \underbrace{f \times f \times 3}_{\text{purada ke ekta filter box vabha jai.}}$ generates $(n-f+1)(n-f+1) \times \underbrace{1}_{\text{one image.}}$

purata ke ekta filter box vabha jai.

one image.

Padding

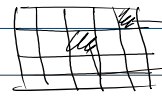
a lil problem in CNN is

→ output size is reduced.

to solve all there we use Padding,

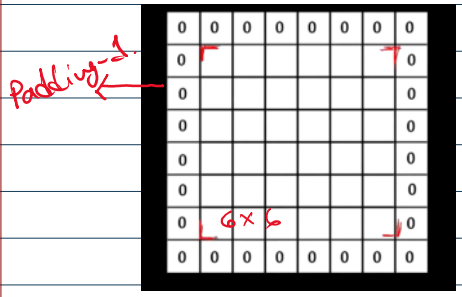
→ 6x6 3x3 → 4x4

→ corner pixels gets less attention.



→ come's under filter once

→ come's " many times.



i. 8x8 3x3 → 6x6 image.

ii. Corner one gets enough attention too.

Incase of applying padding see

i. If no padding is applied (Valid Convolation)

ii. output image should be as input side.

$n=6$, $p=1$ turns out 8.

$n' = n+2p$ ← padding.

or, $n' - f + 1 = n$ → input

output ← $n' - f + 1$

$n+2p-f+1 = n$ → usually taken odd.

$\therefore p = \frac{f-1}{2}$

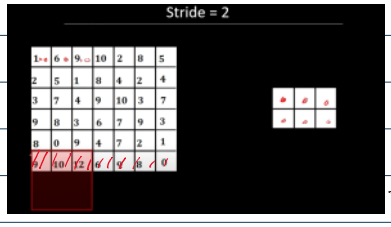
Stride

if we move 1 pixel → strid = 1

" " " 2 " → " = 2.

But what if we don't have enough pixel value in a region?

then we discard that region.



For output size $\left\lfloor \frac{n-f}{s} + 1 \right\rfloor$ → floor value.

↳ stride

$n_1 \times n_2$

$f \times f$

$$\text{output} = \left\lfloor \frac{n_1 - f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n_2 - f}{s} + 1 \right\rfloor = \left\lfloor \frac{6-3}{2} + 1 \right\rfloor \times \left\lfloor \frac{7-3}{2} + \right\rfloor$$

[redacted]

$$\text{output} = \left\lfloor \frac{n_1 - f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n_2 - f}{s} + 1 \right\rfloor = \left\lfloor \frac{6-3}{2} + 1 \right\rfloor \times \left\lfloor \frac{7-3}{2} + \right\rfloor$$

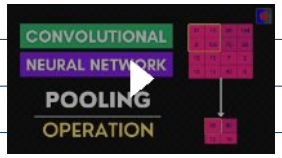
$$= 2 \times 3$$

If padding was used too

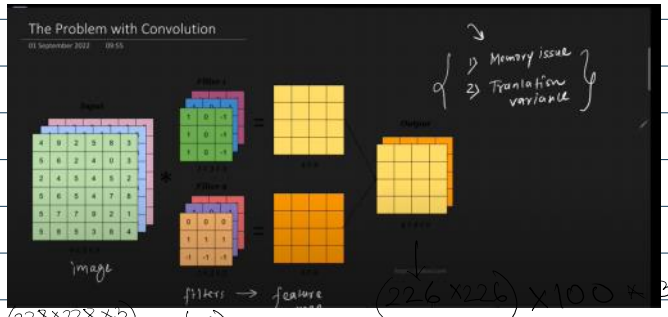
$$\text{then output} = \left\lfloor \frac{n_1 + 2p - f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n_2 + 2p - f}{s} + 1 \right\rfloor$$

Pooling

Pooling Layer in CNN | MaxPooling in Convolutional Neural Network



Till now we have studied



(228×228×3) (100) (226×226)×100×32 → 19 MB storage in RAM (for 1 training data)

If we do it for 100 then → 1.5 GB so large.

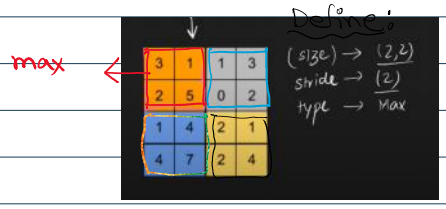
so when convolution is applied er kaj hoitese feature khuja. For example a cat. But problem arrises when cnn look for features it becomes location dependent. means jodi location change kora hoi, are chinbe na. So, we need translation invarience je bical er kan jekhane ei thakuk, oita bical er kan.

For the we need to down sample, your feature map.

⇓

we use pooling.

In order to apply we need a feature map first.



5	3
7	4

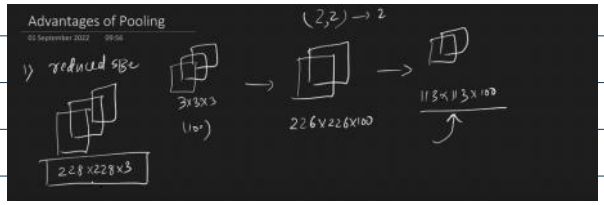
because stride 2.

Basically in a small receptive field, it is keeping the dominant features and eliminating the low-level details.

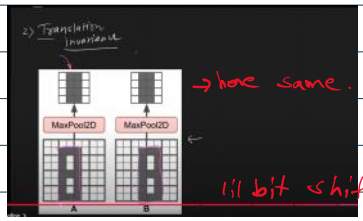
By this translation invarience comes as I down sampled.

Advantage:

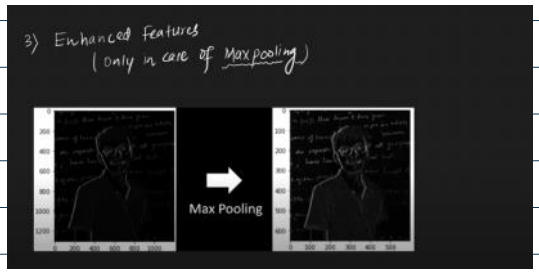
i)

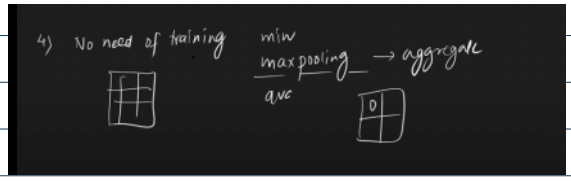


ii)



iii)

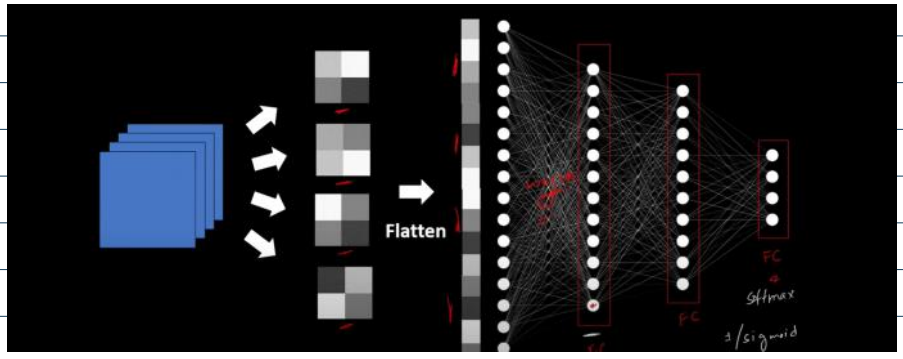




Disadvantage:-

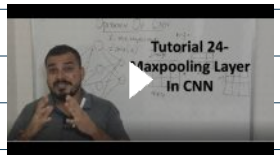
→ location laghle, sad!!

Fully Connected Layers



CNN Part 3 | Convolution Operation

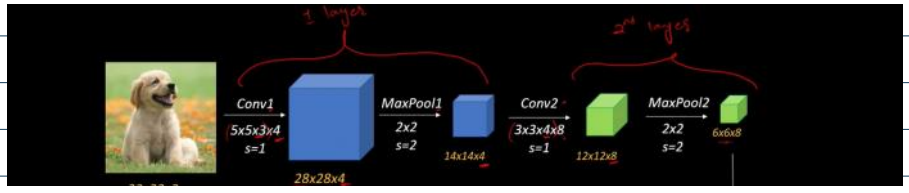
Tutorial 24- Max Pooling Layer In CNN



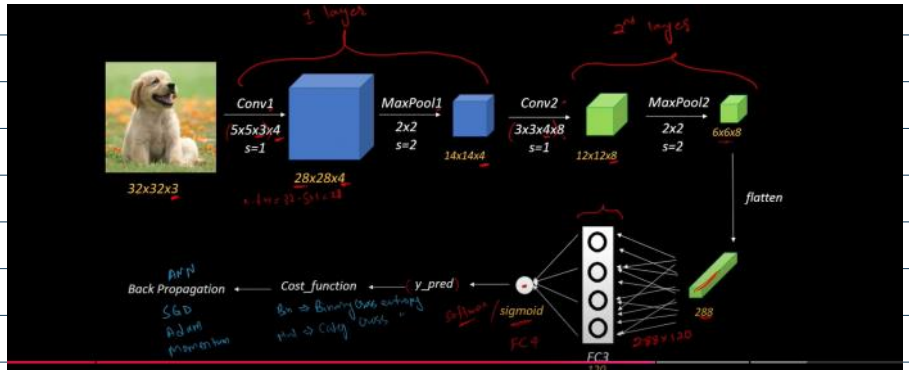
CNN architecture | Explaining the Architecture of CNN

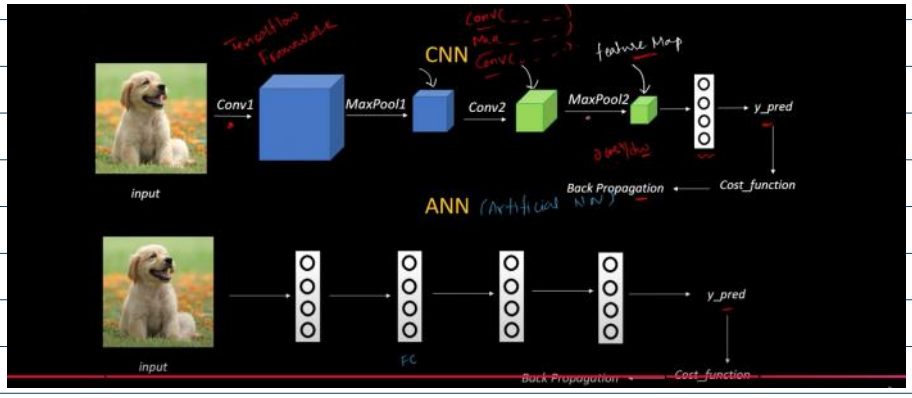
"CNN Architecture"

CNN architecture | Explaining the Architecture of CNN



CNN
Architecture


Conv1
5x5x3x4
s=1
32x32x3
28x28x4
MaxPool1
2x2
s=2
14x14x4
Conv2
3x3x4x8
s=1
12x12x8
MaxPool2
2x2
s=2
6x6x8
flatten
288
FC3
sigmoid
y_pred
Cost_function
Back Propagation


CNN
Conv1
MaxPool1
Conv2
MaxPool2
y_pred
input
Back Propagation
Cost_function
ANN
input
FC
y_pred
Back Propagation
Cost_function

Ann
Friday, December 6, 2024 2:30 PM

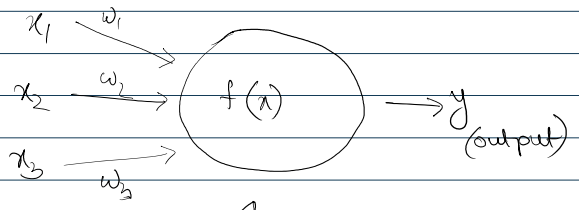
"Neutral Network"

Neutral Network
neurons connected by

Brain → learn.
→ can train itself
→ test

Normal Comp! –
Input Data —Algo→ Meaning —Neutral Network→ Learning & Improvement

ANN → same korte chai.

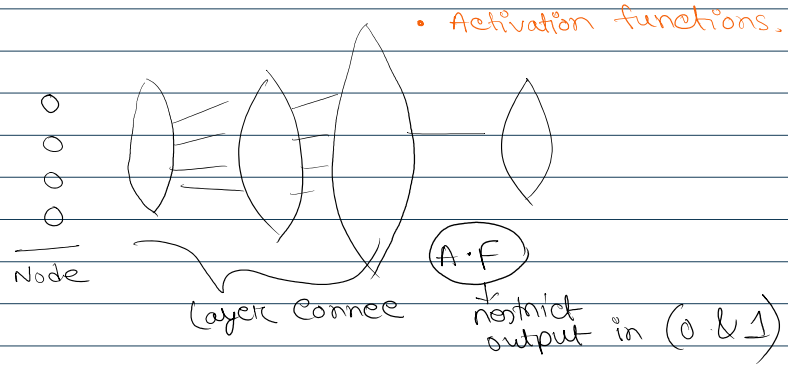


w → primary parameters that a neutral network learns during training.

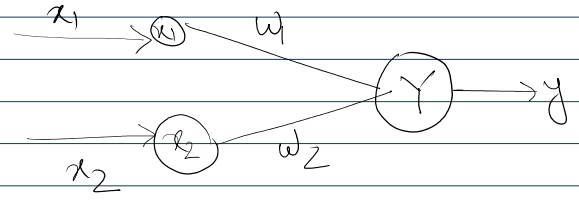
Σ of the function, any function can be

ANN ⇒
Basic component of ANN ⇒

- Neurons (Node) → data store thake
- Layers
- Connections
- Activation functions.



Example! –



Here,
x_1 → total hours a person study
x_2 → t·h·a·p sleeps

$w_1 = 0.5$
$w_2 = -0.3$
$b_1 = 0.1$

lets say we are going to generate output by this way, $y = x_1 w_1 + x_2 w_2 + b$. But we might have "onek chorano ohitano" value. so, we can use different activation function by restrict our output in 0, 1. such a A·F is σ (sigmoid).

chitano" value. so, we can use different activation function by restrict our output in 0, 1. such a A.F is σ (sigmoid).

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

eije w_1, w_2 and b eta time by time change hoi, hote hote ekta state e ashe. Part of learning.

ekta compare korli, bio neutral net are ann er modhe →.

→ dendrite (carry impulse) | → $x_1, x_2 \ldots x_n$

→

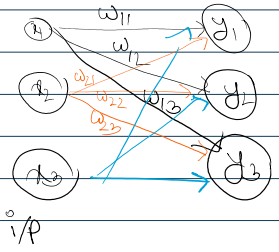
L81: Types of Artificial Neural Network(ANN) Architectures | Artificial Intelligence Course Hindi

Network Architecture

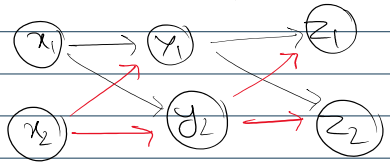
- Single layer feed forward

(input dilam output ashbe) majhe no layer.

⇒ two layers



- Multi layer feed forward (majhe hidden layer)



→ accuracy beshi but shamle jai sudhu.

- Recurrent Neutral Network.

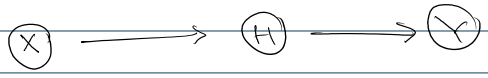
→ memory ache

→ current input state + previous state o ache.

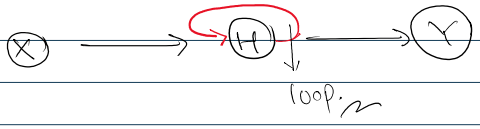
→ basically some add on's on feed forward network.

Recurrent Neural Network (RNN) Part-1 Explained in Hindi

Feed Forward →

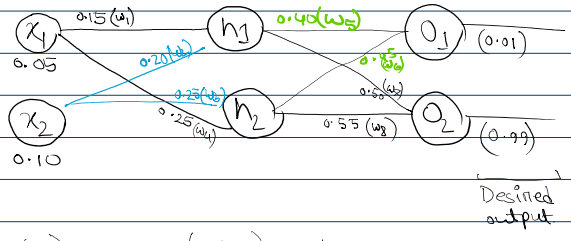


Recurrent →



Back Propagation

$b_1 = 0.35$ $b_2 = 0.60$



going forward,

$h_1(in) = w_1 x_1 + w_2 x_2 + b_1$

$= 0.377$

Now,

$O_1(in) = w_5 \times h_1(out) + w_6 \times h_2(out) + b_2$

eeeeeeeee

$= 0.377$

Desired output

$$h_1(out) = \underbrace{\frac{1}{1+e^{-h_1(in)}}}_{\text{activation function(sigmoid)}} = 0.5932$$

Similarly $h_2(out) = 0.5968$

Now,

$$O_1(in) = w_5 \times h_1(out) + w_6 \times (h_2(out)) + b_2$$
$$= 1.105$$

$$O_1(out) = \frac{1}{1+e^{-O_1(in)}} = 0.7513$$

Similarly $O_2(out) = 0.7729$

checking error,

$$E_t = \frac{1}{2}(target - output)^2$$

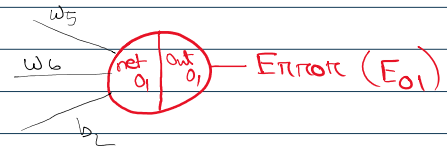
for $E_{o_1} = 0.274$ $\quad E_{o_2} = 0.0235$

error is basically area ki chai on ki ashche tar gap. find at lone we will adjust the parametres to get our desired output.

$$E_{total} = E_{o_1} + E_{o_2}$$
$$= 0.2983$$

going backward

→ adjusting (w_5, w_6, w_7, w_8)



Lets adjust w_5 first,

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o_1}} \times \frac{\partial out_{o_1}}{\partial net_{o_1}} \times \frac{\partial net_{o_1}}{\partial w_5}$$

$$= 0.741365 \times 0.186815692 \times 0.59326$$

$$= 0.08216704$$

$$* \frac{\partial E_{total}}{\partial out_{o_1}} = Out_{o_1} - target_{o_1}$$
$$= 0.751365 - 0.01 = 0.741365$$

$$* \frac{\partial out_{o_1}}{\partial net_{o_1}} = out_{o_1}(1 - out_{o_1})$$
$$= 0.186815692$$

$$* \frac{\partial net_{o_1}}{\partial w_5} = out_{h_1} = 0.59326992$$

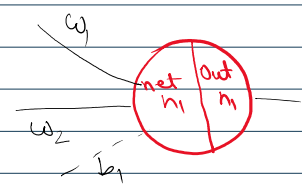
$$New\ w_5 (w_5^*) = w_5 - \alpha \times \frac{\partial E_{total}}{\partial w_5}$$

α = learning rate, assumed (0.6)

$$= 0.4 - 0.6 \times 0.08216704$$
$$= 0.350699776.$$

Similarly calculate w_6, w_7, w_8.

Now, hidden layer there weights (w_1, w_2, w_3, w_4)



Lets find w_1,

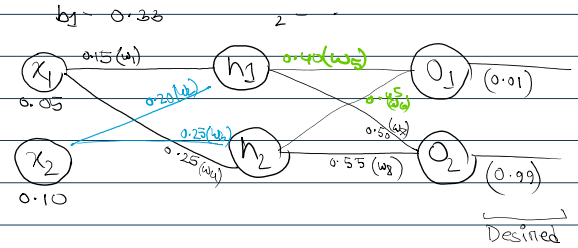
$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out\ h_1} \times \frac{\partial out h_1}{\partial net h_1} \times \frac{\partial net h_1}{\partial w_1} \quad \text{—— (i)}$$

$$* \frac{\partial E_{total}}{\partial out h_1} = \frac{\partial E_{o_1}}{\partial out h_1} + \frac{\partial E_{o_2}}{\partial out h_1}$$

$[E_{total} = E_{o_1} + E_{ou_2}]$

$b_1 = 0.33$

$$* \frac{\partial E_{total}}{\partial out_{h_1}} = \frac{\partial E_{o_1}}{\partial out_{h_1}} + \frac{\partial E_{o_2}}{\partial out_{h_1}} \qquad \left[E_{total} = E_{o_1} + E_{ou_2}\right]$$



$$\frac{\partial E_{o_1}}{\partial net_{o_1}} \times \frac{\partial net_{o_1}}{\partial out_{h_1}} + \frac{\partial E_{o_2}}{\partial net_{o_2}} \times \frac{\partial net_{o_2}}{\partial out_{h_1}}$$

$$\frac{\partial E_{o_1}}{\partial out_{o_1}} \times \frac{\partial out_{o_1}}{\partial net_{o_1}}$$

$= 0.13849856.$

out hidden h_1 to in hidden o_1 edge ta w_5

$$\frac{\partial E_{o_2}}{\partial out_{o_2}} \times \frac{\partial out_{o_2}}{\partial net_{o_2}}$$

$= -0.0380982$

w_7

$$* \frac{\partial E_{o_2}}{\partial out_{o_2}} = (out_{o_2} - target_{o_2})$$

$= 0.7729284 - 0.99$

$= -0.21 \times 0.71535$

$$* \frac{\partial out_{o_2}}{\partial net_{o_2}} = out_{o_2}(1 - out_{o_2})$$

$= 0.17551005 2$

$= 0.036350306.$

$$* \frac{\partial out_{h_1}}{\partial net_{h_1}} = out_{h_1}(1 - out_{h_1})$$

$= 0.241300709$

$$* \frac{\partial net_{h_1}}{\partial w_5} = x_1 = 0.05$$

$$\frac{\partial E_{total}}{\partial w_5} = 0.000438568.$$

$$w^* = w_1 - \alpha * \frac{\partial E_{total}}{\partial w_1} \qquad (\alpha = 0.6)$$

$= 0.149736859 2$

Back Propagation Algorithm (Part-4) Explained with Solved Example in Hindi

Local Search

Sunday, December 8, 2024 9:48 PM

What is local search?

⇒ it searches only the goal state, not the path to get there.

⇒ it does not keep any track of the states/path.

⇒ not systematic.

In local search:

→ hill climbing Algo

→ Simulated Annealing

→ Local Beam Search

→ Evolutionary Algo – Genetic Algo.

Local search diye we can find local maxima. Example – hill climbing.

each iteration e ek thu better solⁿ pawa jai.

But this local maxima/local ideal solution isn't the global or the best solution in all cases.

To overcome that limitation advanced local search came called Metaheuristics.

⇓

global solution/maxima.

Techniques :-

→ Genetic Algo

→ Tabu Search

→ Simulated Annealing

→ Ant Colony Optimazition.

Init

→ mostly nature inspired Algo's.

One of them is

Evolutionary Algo.

Differential Evolution (DE) belongs to a broader category of optimization techniques known as **Evolutionary Algorithms (EAs)**. Let's break down what Evolutionary Algorithms are and the types within this class, including DE, **Genetic Algorithms (GA)**, **Evolutionary Strategies (ES)**, and **Evolutionary Programming (EP)**.

What are Evolutionary Algorithms (EAs)?

- **Definition**: EAs are **population-based** optimization methods inspired by the principles of natural selection and biological evolution.
- **Core Concept**: These algorithms iteratively evolve a set of candidate solutions (population) toward better solutions using mechanisms anabgous to biological evolution (e.g., mutation, crossover, and selection).
- **Key Features**:
  - **Population-Based**: Work with multiple solutions simultaneously.
  - **Randomness**: Introduce randomness in mutation and recombination.
  - **Selection**: Use fitness-based selection to guide the search toward optimal solutions.

Types of Evolutionary Algorithms

1. **Differential Evolution (DE)**:
   - **Focus**: Optimizing real-valued parameters.
   - **Mechanism**: Uses **mutation** (differential variations between vectors), **crossover**, and **selection**.
   - **Strengths**: Effective for continuous, high-dimensional optimization problems.
2. **Genetic Algorithms (GA)**:
   - **Focus**: General-purpose optimization across various domains.
   - **Mechanism**: Mimics natural selection using **chromosome-like representations** (binary strings or real-valued vectors).
     - **Operators**:
       - **Selection**: Fitness-based selection of parents.
       - **Crossover**: Combining parts of two parents to produce offspring.
       - **Mutation**: Randomly changing some elements of the offspring.
   - **Strengths**: Versatile for discrete, combinatorial, and continuous problems.
3. **Evolutionary Strategies (ES)**:
   - **Focus**: Continuous optimization of real-valued parameters.
   - **Mechanism**: Similar to GA but emphasizes **self-adaptive mutation rates** and typically uses **Gaussian mutations**.
   - **Variants**: Often denoted as (μ+λ)(\mu + \lambda)(μ+λ)-ES or (μ,λ)(\mu, \lambda)(μ,λ)-ES, where:
     - **μ\muμ** = Number of parent solutions.
     - **λ\lambdaλ** = Number of offspring solutions.
   - **Strengths**: Effective for fine-tuning continuous parameter spaces.
4. **Evolutionary Programming (EP)**:
   - **Focus**: Evolving the parameters of finite state machines or prediction models.
   - **Mechanism**: Similar to ES but does not use recombination (crossover); focuses on **mutation and selection**.
   - **Strengths**: Ideal for machine learning, behavior optimization, and environments requiring adaptability.

Comparison of the Algorithms

Feature	Differential Evolution (DE)	Genetic Algorithms (GA)	Evolutionary Strategies (ES)	Evolutionary Programming (EP)
Focus	Real-valued parameter optimization	General-purpose optimization	Real-valued parameter optimization	Model adaptation and behavior learning
Main Operators	Mutation (differential), Crossover	Crossover, Mutation	Mutation (Gaussian), Self-adaptation	Mutation
Recombination	Yes	Yes	Limited	No
Application	Continuous optimization	Discrete and continuous optimization	Fine-tuning of real-valued parameters	Adaptive systems and finite state models

Summary

- **Differential Evolution (DE)** is one type of **Evolutionary Algorithm**.
- Other key types include **Genetic Algorithms (GA)**, **Evolutionary Strategies (ES)**, and **Evolutionary Programming (EP)**.
- All these algorithms use principles of evolution (selection, mutation, and recombination) to search for optimal solutions.
- The choice between them depends on the nature of the problem (continuous, discrete, or adaptive).

So, DE:

→ random-ness based, population based Algo.

→ Developed to optimise real value parametre / real valued functions.

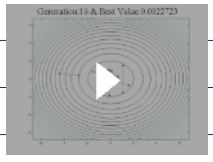
→ Why use Differential Evolution?

• Global optimisation is necessary in fields such as engineering, statistics and finance

• But many practical problems have objective functions that are non-differentiable, non-continuous, non-linear, noisy, flat, multi-dimensional

or have many local minima, constraints or stochasticity
• Such problems are difficult if not impossible to solve analytically
• DE can be used to find approximate solutions to such problems

Differential Evolution Algorithm



Here,

DE	vs	GA
→ real value parametre		→ general valued.
→ only continuous		→ both discrete & continous
→ good for problems where		→ wide range

→ good for problems where parametres are continuous → wide range

→ Mutation, Crossover, Selection → Crossover, mutation, selection.

https://www.youtube.com/watch?v=kyh3Vk10Js4&list=PLVLAu9B7VtkaX0P_m7u8R1_eMqv6hLv0l

Lec 30: Metaheuristic: An Introduction

General Procedure of Evolutionary Algo:-

Notation →

Parametre's = D (real)

Population size, N = 4 (at least 3)

Parametre's vector form $x_{i,G} = [x_{1,i,G}, x_{2,i,G}, \ldots x_{D,i,G}]$

$i = 1, 2, \ldots N$

G = generation number

Step 1 :- (Initialization)

- Define upper and lower bound for each parametre.

example:- let's say we have D=2, means 2 parametres.

- For x_1 → L = −5, → U = 5
- For, x_2 → L = −3, → U = 3

$x_{i,i,G}$ → gen
↳ index of the individual in a population.
parametre → value

- Randomly select initial parametre value's. Say we had N=4 then,

  - $x_{1,1,1} = [1.2, -2.3]$
  - $x_{2,1,1} = [-3.5, 0.8]$
  - $x_{3,1,1} = [4.5, -1.2]$

dutta parametre dhorsi,
x_1 range − [−5, 5]
x_2 " − [−3, 3]

x_1 range $= [-5, 5]$

$x_2 \sim -[-3, 3]$

$x_{1,1,1} = [1.2, -2.3]$

$\downarrow x_1 \quad \downarrow x_2$

3 ta parameter nile

$x_{1,1,1} = [1.2, -2.3, ___]$

$\downarrow x_1 \quad \downarrow x_2 \quad \downarrow x_3$

$x_{2,1,1} = [-3.5, 0.8]$

$\cdot\ x_{3,1,1} = [4.5, -1.2]$

$\cdot\ x_{4,1,1} = [-1.1, 2.2]$

- Mutation:-

each of N parameter goes under mutation, Recombination Selection.

→ mutation expands the search space.

↓

adds a new donor vector. $(v_i, G+1)$

$$v_{i,(G+1)} = x_{r_1,G} + F\,(x_{r_2,G} - x_{r_3,G})$$

F ↓ mutation factor, say 1.5 → usually inbt^n [0–2]

$$v_{i,G+1} = [-2.55, 3.45]$$

↓ donor vector.

Assume randomly,

$x_{r_1,G} = [4.5, -1.2]$

$x_{r_2,G} = [-3.5, 0.8]$

$x_{r_3,G} = [1.2, -2.3]$

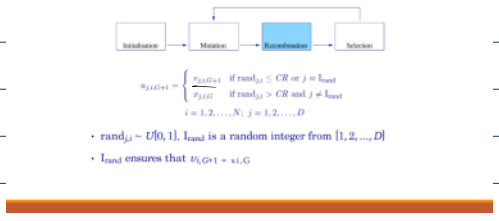
Recombination:

→ it form a Trail Vector

↓ target vector ↓ donor vector.

Recombination combines successful solutions from the previous generation (target vector **xi,G**) with the donor vector **vi,G+1** to form a **trial vector ui,G+1**.

- Recombination incorporates successful solutions from the previous gen - eration
- The trial vector ui,G+1 is developed from the elements of the target vector, xi,G, and the elements of the donor vector, vi,G+1
- ~~Elements of the donor vector enter the trial vector with~~ probability CR



$$u_{j,i,G+1} = \begin{cases} v_{j,i,G+1} & \text{if } \mathrm{rand}_{j,i} \le CR \text{ or } j = I_{\mathrm{rand}} \\ x_{j,i,G} & \text{if } \mathrm{rand}_{j,i} > CR \text{ and } j \ne I_{\mathrm{rand}} \end{cases}$$

$i = 1, 2, \ldots, N;\ j = 1, 2, \ldots, D$

- $\mathrm{rand}_{j,i} \sim U[0,1]$, I_{rand} is a random integer from $[1, 2, \ldots, D]$
- I_{rand} ensures that $u_{i,G+1} = x_{i,G}$

Assume,

$x_{i,G} = [1.2, -2.3] \rightarrow$ target

$v_{i,G+1} = [-2.55, 3.45] \rightarrow$ donor.

$CR = 50\% \rightarrow 0.5$

Now, $\mathrm{rand}_{j,i}$ (bet^n 0 & 1) ekta number nibo, say 0.3 nilam.

As $0.3 < 0.5$ we will take x_1 from $v_{i,G+1}$

Again, $\mathrm{rand}_{j,2} = 0.7 \Rightarrow x_2$ from $x_{i,G}$.

trail vector $= [-2.55, -2.3]$

Randomly, for each element of the parameter vector:

- If a random number **randj,i** (uniformly distributed between 0 and 1) is less than **CR**, take the element from **vi,G+1**.
- If **randj,i** is greater than **CR**, take the element from **xi,G**.

Assume:

- **xi,G = [1.2, -2.3]** (target vector)
- **vi,G+1 = [-2.55, 3.45]** (donor vector)

After applying crossover with **CR = 0.5**, we get:

- **randj,1 = 0.3**, so we take **x1** from **vi,G+1**.
- **randj,2 = 0.7**, so we take **x2** from **xi,G**.

The **trial vector** becomes:

trail vector = [-2.55, -2.3]

Assume:
- **xi,G = [1.2, -2.3]** (target vector)
- **vi,G+1 = [-2.55, 3.45]** (donor vector)

After applying crossover with **CR = 0.5**, we get:
- **randj,1 = 0.3**, so we take **x1** from **vi,G+1**.
- **randj,2 = 0.7**, so we take **x2** from **xi,G**.

The **trial vector** becomes:
ui,G+1 = [-2.55, -2.3]

Selection :-

- **Step:** The target vector **xi,G** is compared with the trial vector **ui,G+1**. The one with the better fitness (lower function value in case of minimization) is selected to be part of the next generation.
- **Example:** Assume we are minimizing a function, and the fitness of the vectors is evaluated. Let's say:
  - Fitness of **xi,G = [1.2, -2.3]** is 5.3.
  - Fitness of **ui,G+1 = [-2.55, -2.3]** is 4.8.
    Since **ui,G+1** has a lower fitness value (better solution), **ui,G+1** will replace **xi,G** in the population.

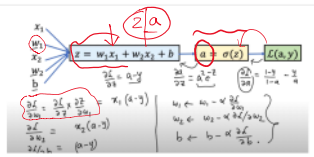
• Mutation, recombination and selection continue until some stopping cri-terion is reached

. **Iteration and Stopping Criterion**
- This process repeats, evolving the population over generations until a **stopping criterion** is met. The stopping criterion could be:
  - A set number of generations.
  - Convergence to an optimal solution (minimal fitness).
  - Satisfying a threshold for improvement in fitness.

Example Summary:
Let's assume we are optimizing a function with 2 parameters and we apply the steps outlined:
- **Initialization:** Randomly generate 4 parameter vectors within bounds.
- **Mutation:** Modify one of the vectors using the mutation formula.
- **Recombination:** Create a trial vector using elements from the target and donor vectors based on the crossover probability.
- **Selection:** Choose the better vector between the target and trial for the next generation.

Crossover in Genetic Algorithm | Methods of Crossover | Reproduction in Genetic Algorithm | GA

Wednesday, December 11, 2024 11:37 PM



$\mathcal{L} = -[y\log(a) + (1-y)\log(1-a)]$

$\frac{\partial(L)}{\partial w_1} = \frac{\partial(L)}{\partial a} \cdot \frac{\partial a}{\partial z} \cdot \frac{\partial z}{\partial w_1}$

$\frac{\partial \mathcal{L}}{\partial o_1} = out_{o_1} - t_1$

$\frac{\partial out_{o1}}{\partial sum_{o1}} = out_{o1}(1 - output_{o1})$

$\frac{\partial sum_{o1}}{\partial w_5} = output_{h1}$

$z = w_1x_1 + w_2x_2 + b$

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

$w_1^* = w_1 - \alpha \cdot \frac{\partial(L)}{\partial w_1}$

$\frac{\partial(L)}{\partial a} \cdot \frac{\partial a}{\partial z} \cdot \frac{\partial z}{\partial w_1}$

$+\left[y \cdot \frac{1}{a} + \frac{1-y}{1-a}\right] = \frac{-a + ay + y - ay}{a(1-a)}$

$= -\left[\frac{y}{a} + \frac{1-y}{1-a}\right] = \frac{a-y}{a(1-a)}$

$\frac{\partial a}{\partial z} = a(1-a)$

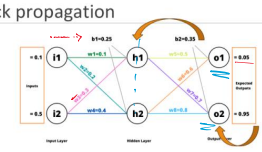
$\frac{\partial z}{\partial w_1} = x_1$

$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial(L)}{\partial z} \times \frac{\partial z}{\partial w}$

Combine

$= \frac{\partial \mathcal{L}}{\partial w_1} = x$

Back propagation



Forward pass → relu, $= \max(0, x)$

$\Sigma \mid 0$

h

$sum\,h_1 = i_1w_1 + i_2 \times w_3 + b_1$ | $out\,h_1 = \frac{1}{1+e^{-sum_1}} = 0.60$

$= 0.41$

$sum\,h_2 = i_1 \times w_2 + i_2 \times w_4 + b_2$ $out\,h_2 = 0.61538$

$= 0.47$.

Similarly,

$sum_{o_1} = 1.01977$ | $sum_{o_2} = 1.2306$

$out_{o_1} = 0.73492$ | $out_{o_2} = 0.7295$

$\mathcal{E}_{o_1} + \mathcal{E}_{o_2} = \mathcal{E}_{total}$.

$\frac{1}{2}(0.05 - 0.73)^2$

$= 0.234 + 0.014$

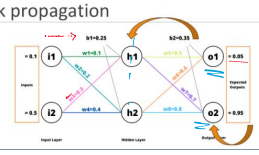
$\mathcal{E}_{t_2} = 0.24908$.

Back P:- w_5

$(new_w5) = w_5 - \alpha \times \frac{\partial(E)}{\partial w_5}$

$\Sigma \mid o_1$

Back propagation



Forward

$sum\,h_1$

$\frac{\partial(E)}{\partial w_5} = \frac{\partial(E)}{\partial o_1} \times \frac{\partial out_{o1}}{\partial sum_{o1}} \times \frac{\partial sum_{o1}}{\partial w_5}$.

$\frac{\partial(E)}{\partial o_1} = \frac{\partial}{\partial out_{o_1}}\left\{\frac{1}{2}(t_1 - out_{o1})^2 + \frac{1}{2}(t_2 - out_{o2})^2\right\}$

$\frac{\partial(E)}{\partial out_{o_1}} = \frac{\partial}{\partial out_{o_1}}$

$\frac{\partial \mathcal{L}}{\partial o_1} = 0$

$(a-y)$.

$\omega t_{01} - t_1$

$$\frac{\partial(E)}{\partial out_{o_1}} = \frac{\partial}{\partial out_{o_1}} \frac{1}{2}(t_1 - out_{o_1})^2$$

$$= \frac{1}{2} \times 2(t_1 - out_{o_1})(-1)$$

$$= \frac{1}{2} \times 2(out_{o_1} - t_1)$$

$$= \boxed{out_{o_1} - t_1}$$

C-2

$$\frac{\partial(out_{o_1})}{\partial sum_{o_1}} = \frac{\partial}{\partial sum_{o_1}} \left(\frac{1}{1+e^{-sum}}\right)$$

$$= output_{o_1}(1 - output_{o_1})$$

$$\frac{\partial E}{\partial o_1} = 0$$

$$\frac{\partial out_{o_1}}{\partial sum_{o_1}} =$$

$$\frac{\partial sum_{o_1}}{\partial w_5}$$

C-3

$$\frac{\partial sum_{o_1}}{\partial w_5} = \frac{\partial}{\partial w_5}(sum_{o_1})$$

$$= output_{h_1}$$

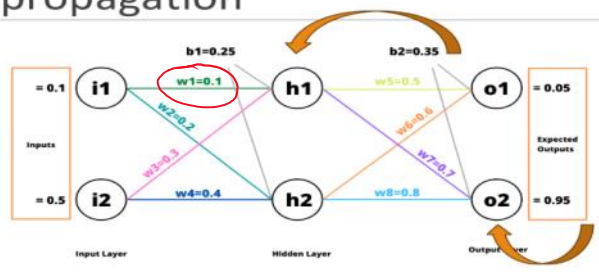
w_5

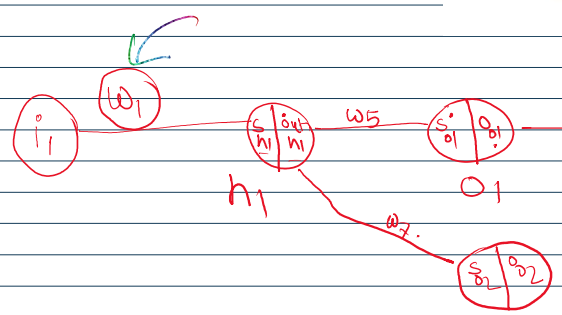
$$\frac{\partial E}{\partial o_1} = out_{o_1} - t_1$$

$$\frac{\partial out_{o_1}}{\partial sum_{o_1}} = out_{o_1}(1 - output)$$

$$\frac{\partial sum_{o_1}}{\partial w_5} = output_{h_1}$$

Back propagation





$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_1}{\partial w_1} + \frac{\partial E_2}{\partial w_1}$$

$$\frac{\partial E_1}{\partial w_1} = \underbrace{\frac{\partial E_1}{\partial out_{o_1}}}_{C_1} \times \underbrace{\frac{\partial out_{o_1}}{\partial sum_{o_1}}}_{C_2} \times \underbrace{\frac{\partial sum_{o_1}}{\partial out_{h_1}}}_{C_3} \times \underbrace{\frac{\partial out_{h_1}}{\partial sum_{h_1}}}_{C_4} \times \underbrace{\frac{\partial sum_{h_1}}{\partial w_1}}_{C_5}$$

C_1

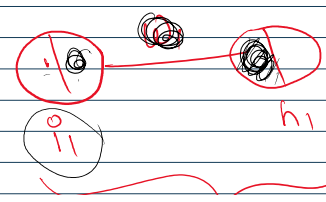
$$\frac{\partial E}{\partial o_1} = out_{o_1} - t_1$$

C_2

$$\frac{\partial out_{o_1}}{\partial sum_{o_1}} = out_{o_1}(1 - output)$$

C_5

$$\frac{\partial sum_{o_1}}{\partial w_5} = output_{h_1}$$



$w_{o_1} - t_1$

$out_{o1}(1 - output_{o1})$

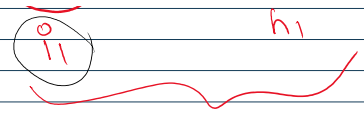
$= output_{h1}$

$$C_2 = \frac{\partial out_{o1}}{\partial sum_{o1}} = out_{o1}(1 - output)$$

$$\frac{\partial sum_{o1}}{\partial w_5} = output_{h1} \leftarrow C_5$$

$$C_3 = \frac{\partial (sum_{o1})}{\partial out_{h1}} = \frac{\partial}{\partial out_{h1}} \left[output_{h1} * w_5 \right] + \cancel{out_{h2} * w_6 + b_2}$$

$$= w_5.$$



$$\frac{\partial sum_{o1}}{\partial w_5} = \cancel{out_{h1}} \; 1.$$