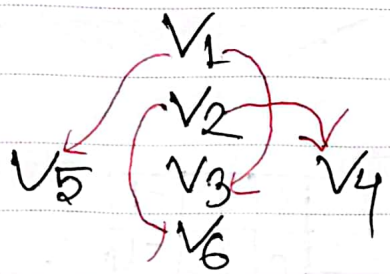
Su Mo Tu We Th Fr

Date:...../ ...../ .....

Difference type of layer in CNN

CNN is a part of deep learing which comes into the picture inspiring from, ~~the~~ how the brain works.

In the back part of our head ~~eh~~ we have Cerebral Cortex and an important portion of this is called Visual Cortex responsible for creating the image in our head.

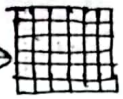


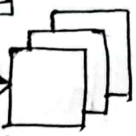
@ After getting data through our sensory organ eye, it pass through various layer presented in the visual cortex area. And all those interconnected layers ~~are~~ do some operation on the image data and, ~~pass~~ And share information among all of the layers and finally add the end image get created. And all all the layers are responsible for detecting

specefic part of the image.

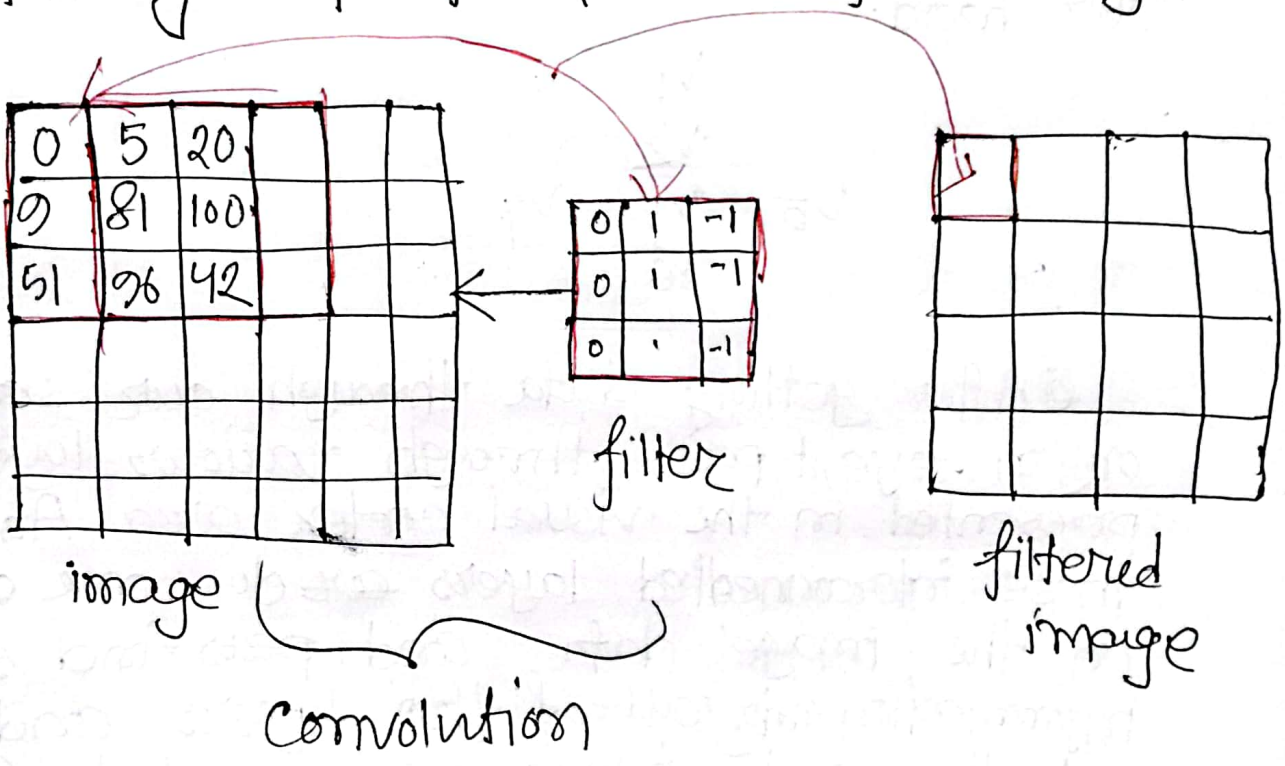
* Convolution

Basically we work with two types of

1) Grey scale (6x6) → 

2) RGB scale (6x6x3) → 

Filter: are mxn matrix responsible for detecting specific portion of an image.



* Padding: It is used to retrieve the corner information

* zero padding
* Neighbour padding.

Sa Su Mo Tu We Th Fr

Date:...../ ...../ .....

* Stride: It means how much steps we will take both vertically & horizontally.

⇒ Equation for getting output shape of an image after filtering.

$$(\text{Width}, \text{height}) = \left\{\left(\frac{W - f_w + 2P + 1}{S}\right), \left(\frac{H - f_h + 2P + 1}{S}\right)\right\}$$

N.B if we want the output size will be same as input,

$$W - f + 2P + 1 = W$$
$$\text{hence, } P = (f-1)/2$$

And here, height & width of the filter will be same.

* How CNN is different from ANN
  * Parameter sharing.
  * All the weights and (Input value or Neuron value) do not ~~must~~ get multiplied to define the value of a neuron in next layer.

* Why do use CNN over ANN for image classification:

* Large numbers of trainable parameter.
* Loss of spatial features.
* Vanishing & exploiding gradient dsecnt.

* What is vanishing Gradient descent:

It occurs when weight updation is less, ~~can be~~ and ignorable amount

$$new = old - \eta * \frac{\delta}{\delta w}$$

$$= 5 - \boxed{.000123} \leftarrow \text{can be ignored}$$

* What is Exploiding Gradient descent:

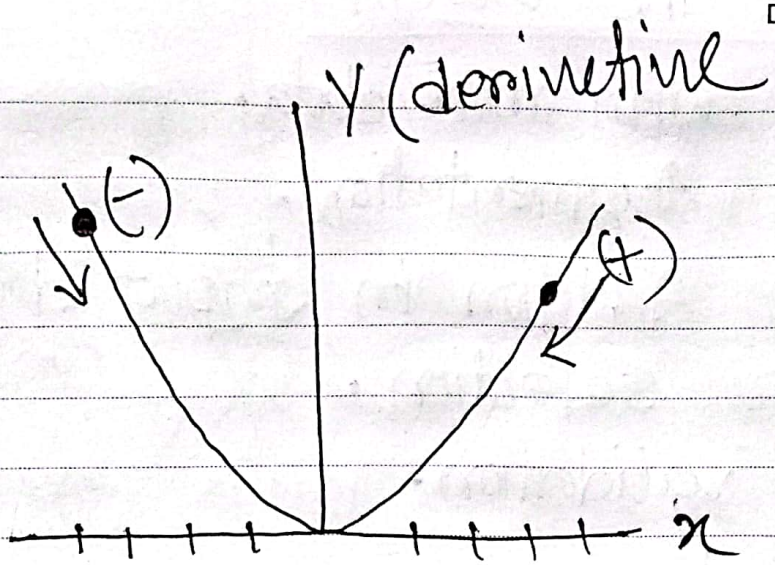
It occurs when, the ~~summal~~ multiplication of derivative of loss function w.r.to weights is to large.

$$new = old - \eta * \frac{\delta}{\delta w} \frac{\delta}{\delta L}$$

$$= 5 - \boxed{4.123} \leftarrow \text{way too much}$$

Sa Su Mo Tu We Th Fr

Date:...../...../.....



Gradient descent Graph.

* Why do we need bias? Go to ML Khata
* Which pooling technique will you use? same

* Overfitting & Underfitting:

Model get underfitted, ~~when~~ in single layer ~~or very less num~~ Neural Network or which has less number of ~~Neura~~ layer. Underfitted model has high ~~variance~~ biar low variance.

Overfitting simply define the memorization of training data. And model will have high bias & low variance.

* To reduce the overfit:

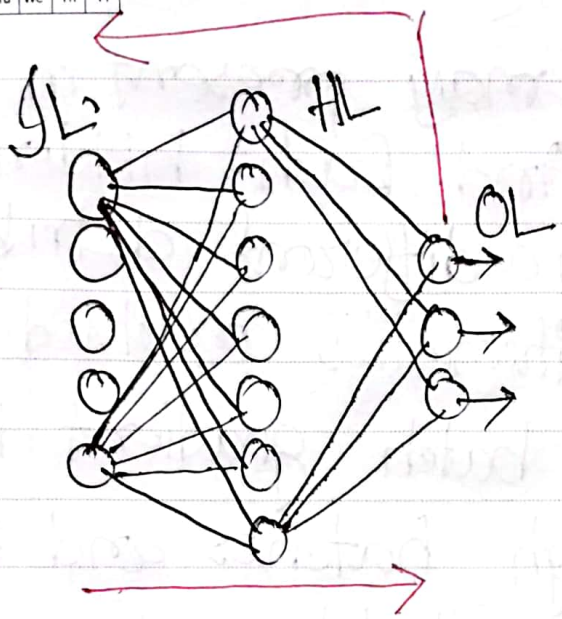
1. Train with more data.
2. Data Augmentation.
3. Noise Addition in Input data.
4. Feature selection.
5. Cross-validation.
6. Model tunning.
   - * Pruning of Decision tree. (Post & Pre)
   - * L1, L2 Regularisation.
   - * Using Dropout in the Model's layers.
7. Ensembling.
8. Early stopping.

* Dropout layers:

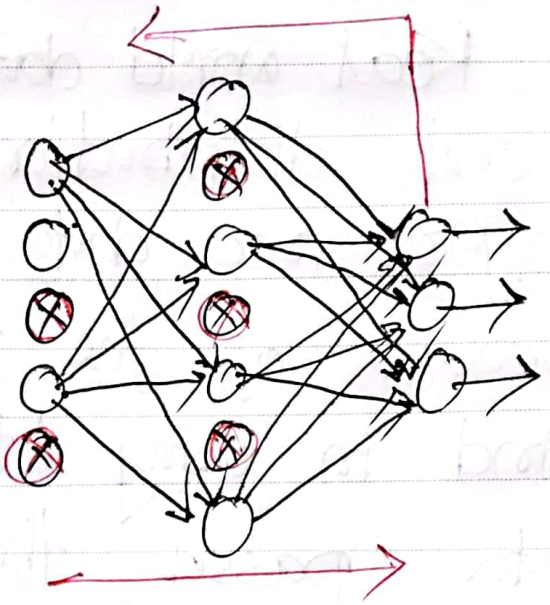
Dropout means randomly removing Neuron from the hidden layers and the Input layer, which contains the features of data. In a word random feature and neuron selection in each epoch.

Sa Su Mo Tu We Th Fr

Date:...../...../.....



Before dropout



After dropout

In the forward propagation randomly a proportion of Neuron will be deactivated and in backpropagation weight corresponding to those neuron will not be updated. And this will continue for the n numbers of epochs.

And while ~~backpropagation~~ & testing model, weights of each layer will by multiplied by the corresponding dropout ratio of that layer

Look like Random forest Algorithm.

* Batch Normalization: (Internal covariant shift)

Real world data may present in different distribution. And Each hidden layers see data in a different distribution because of the mathematics involved in it. And in using Mini batch gradient discent data passes through batches and back propagate for each batch and update the weights. But just because of variety in distribution, it's hard to recognise pattern and takes time to converge. And that's why Batch normalization comes into the picture for normalizing dat in each layer and Keep same distribution throughout the process.

Basic Mathematic of Batch Normalization

(1) Calculate mean value of all the neuron for all the data points and in a batch, as well as the standerd deviation.

Sa Su Mo Tu We Th Fr

Date:...../...../.....

(1) $\mu = \frac{1}{n}\sum z^{(i)}$ (2) $\sigma^2 = \frac{1}{n}\sum (z^{(i)} - \mu)^2$

(2) Then it normalizes the activation vector, in this way all the neuron's output follows a normal distribution

(3) Then finally it calculate the layer's output by applying a linear transformation ~~with~~ by adding two new variables (trainable) α, β to introduce non linearity.

(3) $z^{(i)}_{norm} = \frac{z(i) - \mu}{\sqrt{\sigma^2 - \epsilon}}$ (4) $z = \gamma * z^{(i)}_{norm} + \beta$

* Loss function or Cost function

Loss function is used to define ~~e~~ individual losses & Cost function for the cumulative ~~loss~~ errors.

→ Different types loss function are there

(1) Logloss or Binary Cross Entropy.
Its used in logistic Regression

(2) Sparse-Categorical cross entropy.
(3) Categorical_crossentropy.
(4) Mean Absolute Error.
(5) Mean Squared Error.
(6) Root Mean Square Error.
(7) R^2 squared Error.
(8) Adjusted R^2 squared error.

* Python Interview Question.

what is Zip function.

zip method takes multiple iterable or containers and return a touple(iterator object) having maped values from all the containers

* Few confusing terms in Compnter Vision

Imaga Classification, Objeet Recognition. Objeet detection, Objeet segmentation, Objeet verification.

Sa Su Mo Tu We Th Fr

Date:...../ ...../ .....

⇒ Image Classification/ Image Recognition

Means classifying whole image into a category.

⇒ Image or Object localization:

Means Localizing the main object in the Pictur image alongside the classification. When Image will contain only 1 object.

↳ Object Detection

It specifies the location of Multiple object along with it's class.

⇒ Object segmentation

It's all about segmenting the pixels of an image, having similar attributes. And another important point is it's gives us the shape of the object.

Disopan®

Image Recognition Project

1. Loading Neessary library.
2. Loading the Image
3. Finding the face_location from Actuall image.
4. Getting feature or the encoding of Image.
5. Repeat 3 & 4 for the test image also
6. And compare those image using the face-recognition Library.
7. Also can find the distance between those.

Sa Su Mo Tu We Th Fr

Date:...../ ...../ .....

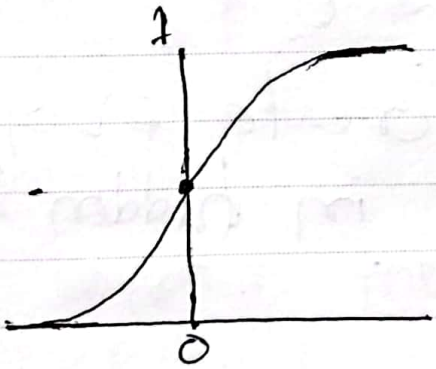
Object detection Algorithm

R-CNN, Fast-CNN, Faster-CNN, YOLO (Have to Read)

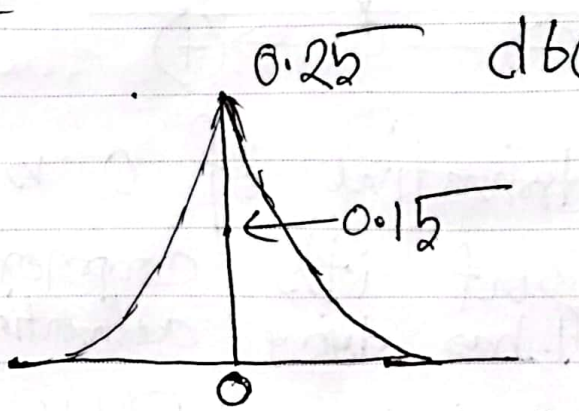
Activation Function

(1) Sigmoid

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



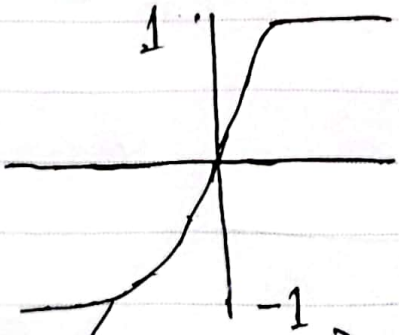
value (0, 1)



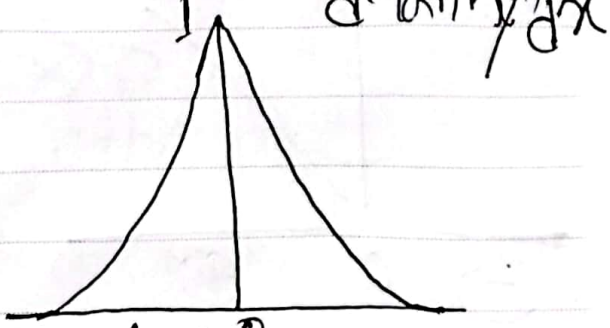
derivative (0, 0.25)

(2) tanh

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



Value (-1, 1)

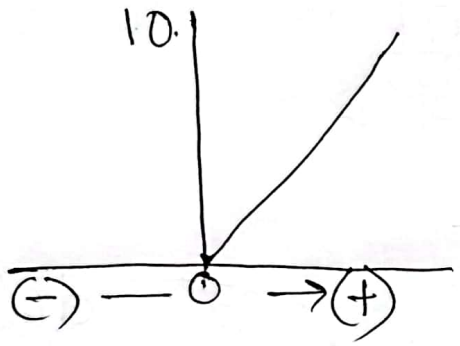


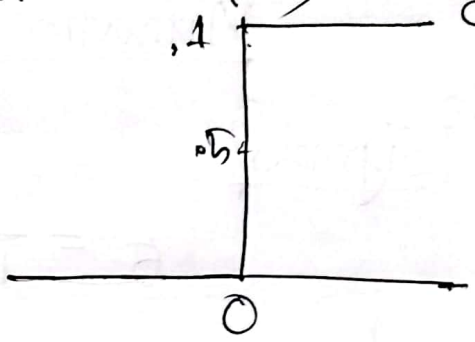
derivative (0, 1)

Above both as vanishing Gradient descent problem,

③ Relu

~~Relu~~ ReLU = max(0, x)



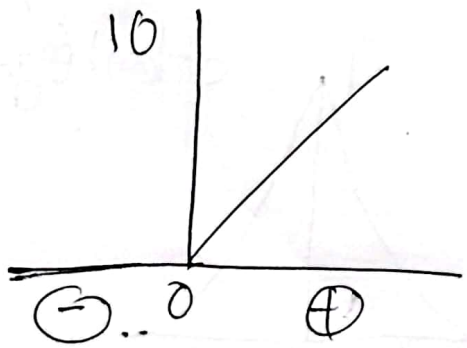


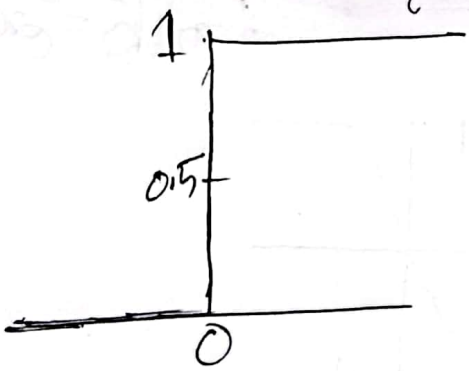
drivative of 0 is 0 and create problem.
But it's commonly used in hidden layer.
It has dying activation problem.

④ Leaky ReLU

$f(x) = \max(0.1x, x)$

$df(x)/dx$





It add a smaller value to negative value to eradicate dying activation problem.

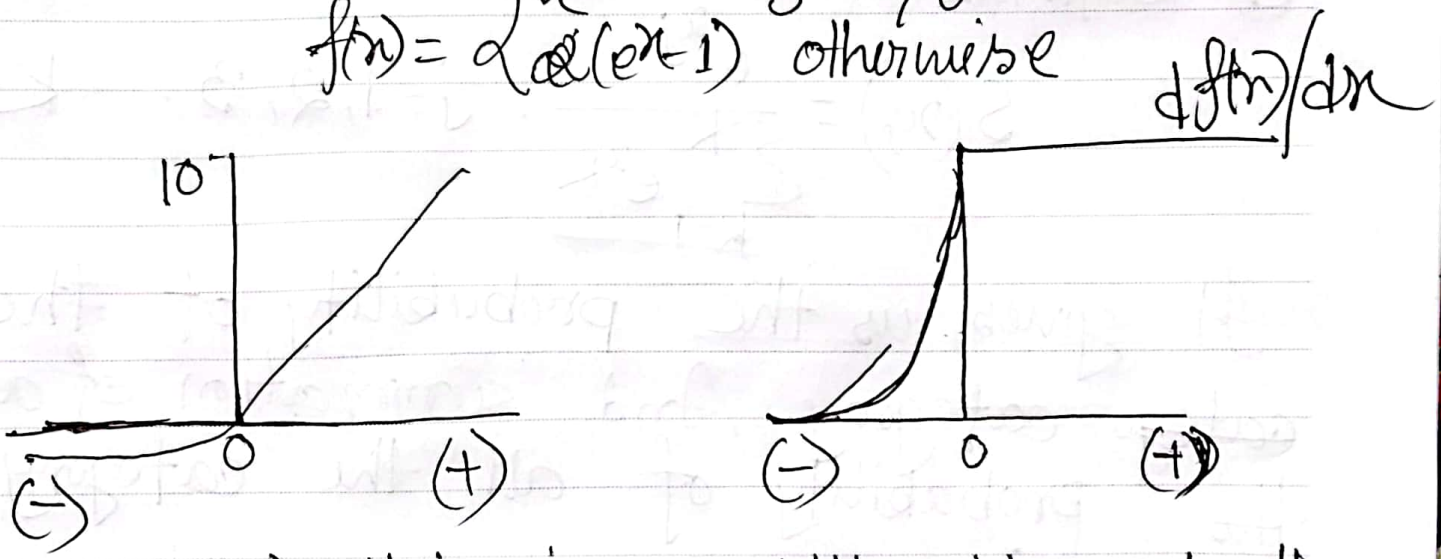
Sa Su Mo Tu We Th Fr

Date:...../ ...../ .....

But Leaky Relu create the vanishing gradient descent problem again

⑤ ELU:

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(e^x - 1) & \text{otherwise} \end{cases}$$



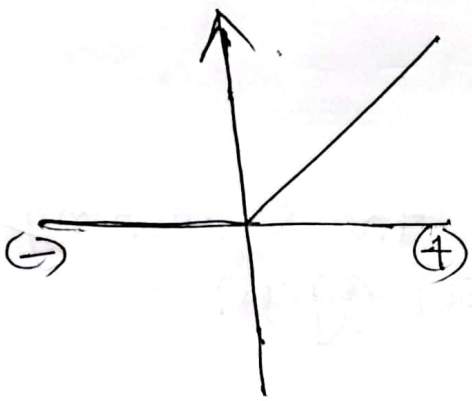
$\alpha(e^x - 1)$ this gives a little higher value than leakyrelu and derivative of it also higher. thus handle the over GD problem.
But it takes time computationaly expensive

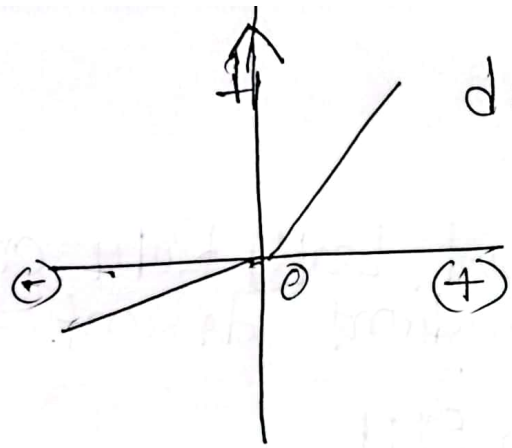
⑥

⑥ PRelu (Parametric Relu):

$$f(x) = \begin{cases} x & x > 0 \\ \alpha x & \text{otherwise} \end{cases}$$

if $\alpha = 0.01$ it becomes LeakyReLU
Here α is learning Parameter
it will learn dynamically while training





⑦ Softmax

$$S(x_j) = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}}, \quad j = 1, 2, 3 \cdots K$$

It gives us the probability of the ~~catego~~ category. And summation of all ~~the~~ probability of all the category is 1.

Assume last layer has value something like this.

[40, 30, 20, 10]

↓

$$P(40) = \frac{e^{40}}{e^{40} + e^{30} + e^{20} + e^{10}}, \quad P(30) = \frac{e^{30}}{e^{40} + e^{30} + e^{20} + e^{10}}$$

Sa Su Mo Tu We Th Fr

Date:...../...../.....

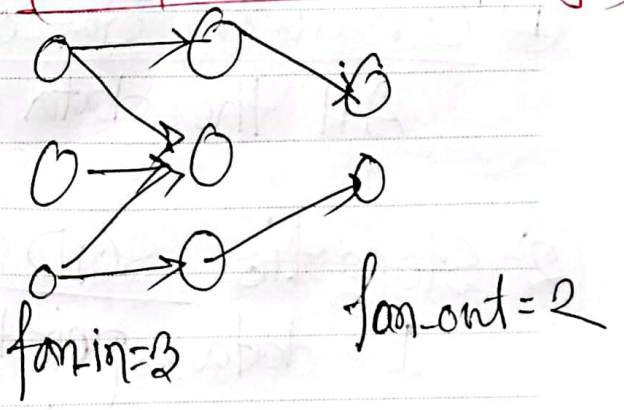
Weight Initialization Technique

1. Weights must be balanced.
2. Weights should not be same.
3. Weights should have good variance.

Various Techniques

① Uniform Distribution: (Works well with Sigmoid)

$$W_{ij} = \left[\frac{-1}{\sqrt{fan\_in}}, \frac{1}{\sqrt{fan\_out}}\right]$$



② Xavier/ ~~Glorot~~ Gorot (Works well with Sigmoid)

① Xavier Normal ② Xavier Uniform

$$W_{ij} \sim N(0, \sigma)$$

$$\sigma = \sqrt{\frac{2}{(fan\_in + fan\_out)}}$$

$$W_{ij} = \left[\frac{-6}{\sqrt{(fan\_in + fan\_out)}}, \frac{6}{\sqrt{fan\_in + fan\_out}}\right]$$

(3) He init (Works well with ReLU activation)

(i) He uniform

$$W_{ij} \approx \left[-\frac{6}{\sqrt{fan\_in}}, \frac{6}{\sqrt{fan\_in}} \right]$$

(ii) He. Normal

$$W_{ij} \approx N(0, \sigma)$$

$$\sigma = \sqrt{\frac{2}{fan\_in}}$$

Optimizers

(1) Gradient descent:
All the data at a time & ~~time killing & take~~ to much resources
take so much resources

(2) ~~Stocastic~~ SGD:
1 data point at a time.
take to much time.

(3) Min batch SGD:
Batch of data, take less time and converge faster but noisy.

(4) SGD + Momentum:
SGD + momentum to remove noise.

(5) ~~Adadelta and RMSprop~~ Adagrad

Sa Su Mo Tu We Th Fr

Date:...../...../.....

6. Adadelta and RMSprop.
7. Adam (RMSprop + Momentum)

Adam is most used Optimizers.

Tensor

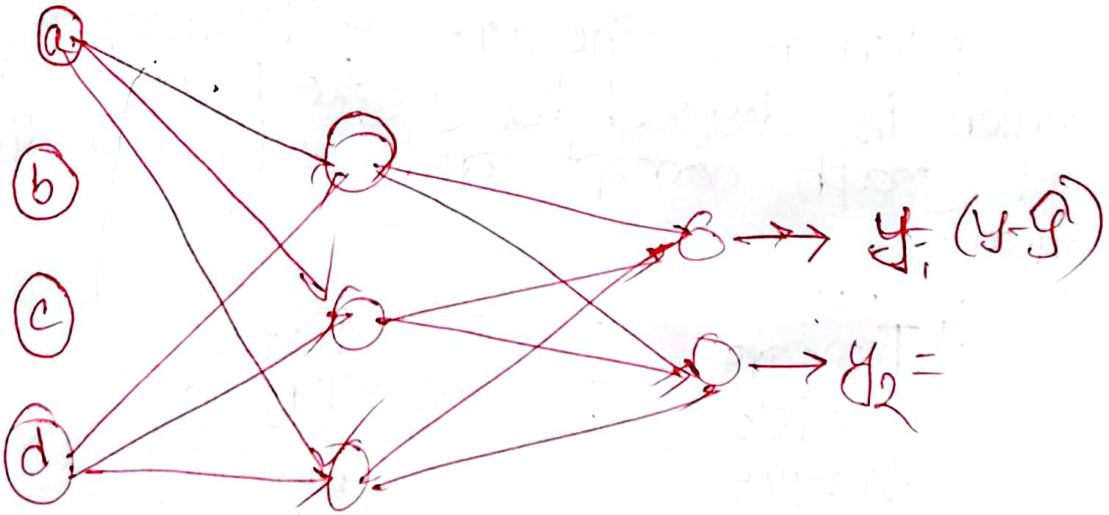
Tensor: Tensor is a vector or matrix of n-dimensional that represent all types of data. All values in a tensor hold identical data type with a known shape. The shape of the data is the dimensionality of the matrix or array.

* Tensor is the core of Tensorflow library, which is designed by google having computational graph concept and automatic differentiation.

Tensors	Data
Scaler	[1] (0D)
Arrays	[1,2,2] (1D)
Matrix	[[1,2,3],[4,5,6]] (2D)
Multidimensional or ndarray / N dimensional tensor	[[[1],[2],[3]],[[1],[2],[3]]] (3D)

?Tensor contains:

1. Input Data.
2. Store the output of set of compution take place inside a graph (Known as Computional graph)
3. Keep the values of weights and bais biases update while backpropagation.
4. And it also take care of back propagation with automatic differentiation also Known as chain rule.



Computation graph

Most Important (Chain Rule)

Interrelated differentiation.

Sa Su Mo Tu We Th Fr

Date:...../...../.....

Cosine Similarity & distance

① Minkowski distance:

$$D = \left(\sum_{i=1}^{N} |P_1 - P_2|^P \right)^{1/P}$$

And from this Equation, can derive Manhatten and Eucledian distance.

Manhatten: When $p = 1$ $= |x_1 - x_2| + |y_1 - y_2|$

Eucledian: When $p = 2$ $= \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$

Both are Good but when features increases may create complexity.

② Haming distance:

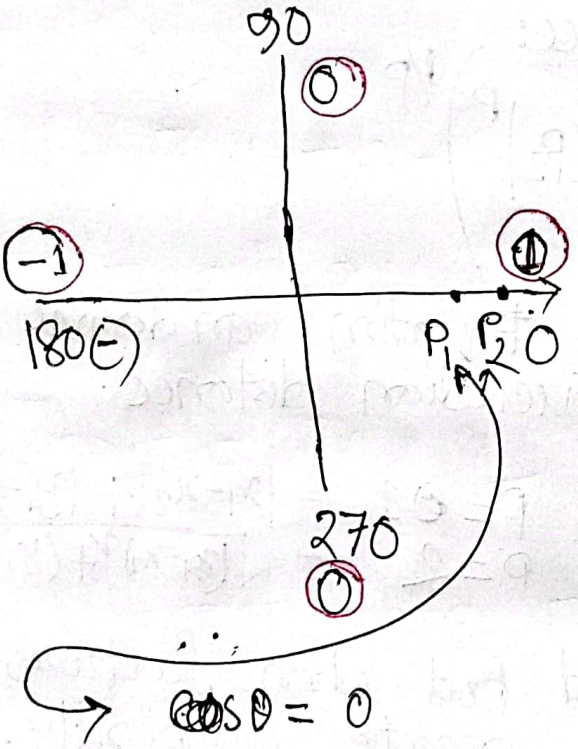
It finds the simmilarity between two equal size strings. And check character by character and do bit bit wise operation

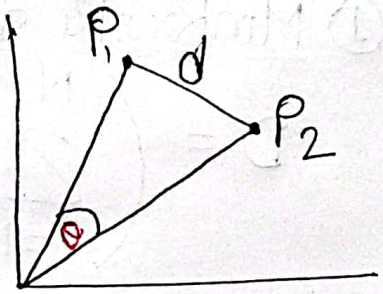
Sudha
Sudda
1 1 1 0 1

So, HD = (1+1+1+1+1) = 4

③ Cosine similarity & Distance:

$1 - \text{Cos\_sim} = \text{Cos\_dis} = \cos\theta$





And the similarity value ranging between (0, 1)

$\cos\theta = 0$

$\because \cos\theta = \cos 0 = 1$

So, this two points are most similare

$$\text{(S) similarity}(A, B) = \frac{A \cdot B}{\|A\| * \|B\|} = \cos\theta$$

$$\Rightarrow \cos\theta = S$$

$$\Rightarrow \theta = \cos^{-1}(S)$$

Sa Su Mo Tu We Th Fr

Date:...../...../.....

Recurrent Neural Network (RNN)

⇒ RNN is a special type of NN deals with Sequential type of data by remembering the previous history, which includes a hidden state to remember the past data.

⇒ But it failed to remembering Long data, and also vanishing gradient descent problem.

⇒ And then came the LSTM which resolves this issue by adding a cell state alongside hidden state to keep important past information, containing three main gates. And hidden state will be for output the result and cell state is for keeping important information.

⇒ Forget Gate: Responsible for deciding which information need to be removed.

$$f_t = \sigma(x_t U_f + h_{t-1} W_f + b_f)$$

And sigmoid activation function is used on top it, ranging between (0~1). (mean remove information.

⇒ @Input Gate: It's responsible for adding new information will be stored in the cell state, with the help of candidate state,

$If = \sigma(x_t u_i + h_{t-1} w_i + b_i)$

$g_t = \tanh(x_t u_g + h_{t-1} \cdot w_g + b_g$

Input $= If * g_t$

⇒ Output Gate: For deciding which information of cell state will goto output, by using tanh on Input

$Cell\text{-}in = \tanh(Input)$

And finally updating hidden state

$h_t = \sigma(x_{th} u_{th} + h_{t-1} w_{th} + b_t) * Cell\text{-}in$

And $\hat{y} = Softmax(h_t)$

And loss $= (y - \hat{y})$ then backpropagating

Sa Su Mo Tu We Th Fr

Date:...../ ...../ .....

⇨ BiLSTM Depends on the Knowledge of bothe left and Right side for prediction

⇨ GRU, BiGRU → (পরে দেখবো অন্য খাতায়)

[Seq to Seq]

- Encoder-Decoder (BLUE Score)
- Attention
- Transformer
- BERT