Machine Learning

* Machine Learning:

→ 3 types.

→ Supervised = Labeled
→ Unsupervised = unlabeled.
→ Reinforcement.

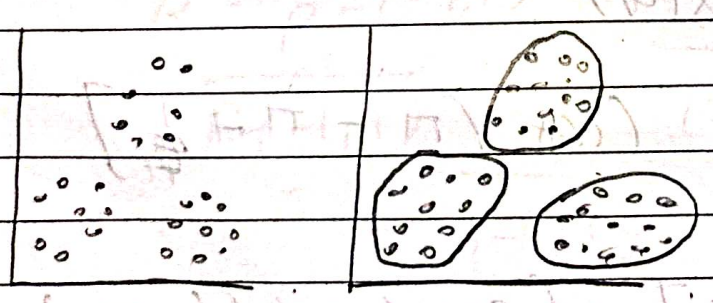
* Unsupervised Learning:

1] Clustering
2] Association.

Clustering Algorithms

1] K-mean | Hierarchical | Mean Shift | Density Based.

No of clusters
if K = 3



Before K-mean | After K-mean.

Inter cluster distance must be minimum
&
Intra cluster distance must be Maximum

2] Hierarchical

bottom-up
→ Agglomerative - each data as individual cluster then merge k clusters

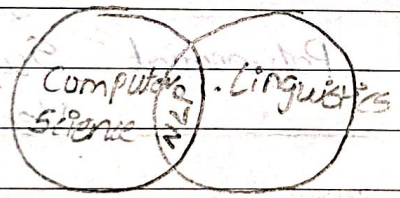
top-bottom
→ Divisive - all data as one cluster then divide to small clusters.

3 types of distance: single (Min), complete (Max) & Average Link.

3] Mean-Shift.
→ It slide to the place where dense data points will be form.
→ Find centroid.

4] Density based.
→ It works based on density.

* Natural Language processing:



→ NLU = Natural language understanding = Understand language
→ NLG = Natural language generation = Generate language

→ Natural language tool kit (NLTK), SCI-KIT Learn

→ 2 NLP techniques

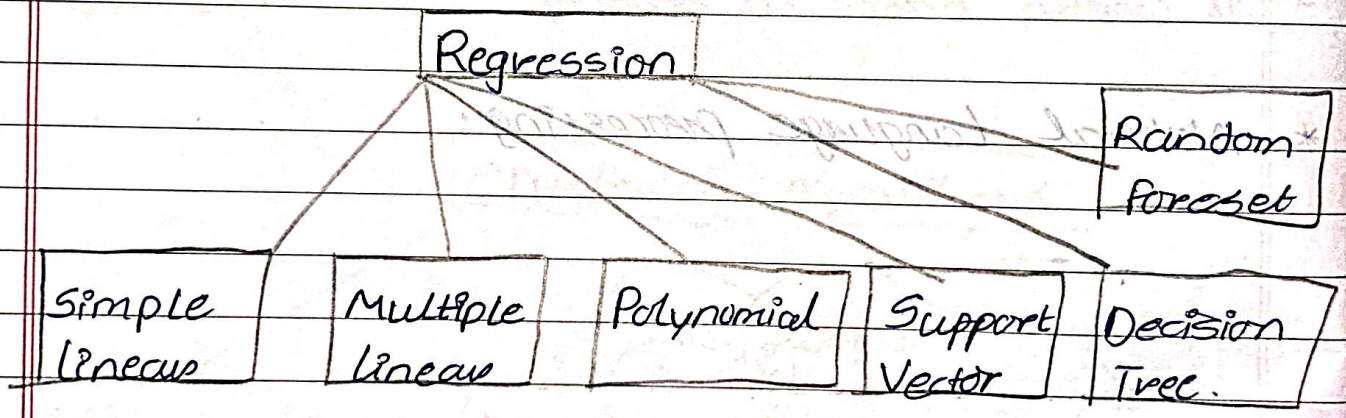
→ Syntactic Analysis = Formal grammer rules in group
→ Semantic Analysis = Understanding & interupting the words, sign, structure of sentence.

* Reainforcement Learning:
→ It works based on good and bad Feedbacks.

1] → Positive
2] Negative

* Supervised Learning:

→ Regression
→ classification



* Classification:

→ Logistic regression
→ K-nearest neighbours
→ Support vector
→ Kernel SVM
→ Naive Bayes
→ Decision Tree classification
→ Random Forest.

classmate

Date
Page

* Linear Regression:

$$h_\theta = \theta_0 + \theta_1 x_1$$
$$y = mx + c$$

We predict the y with the help of slope and the c. Here x is fix

Cost Function

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i)^2$$

m = No of example

→ We have to find the min error from this function.

Gradient Descent Algorithm:

Hyper parameter / learning rate

$$\theta_j = \theta_j - \alpha \underbrace{\frac{\partial}{\partial \theta} J(\theta_0, \theta_1)}$$ ← Gradient

New Value

Old value

$$temp\theta_0 = \theta_0 - \alpha \frac{\partial}{\partial \theta} J(\theta_0, \theta_1)$$

$$temp\theta_1 = \theta_1 - \alpha \frac{\partial}{\partial \theta} J(\theta_0, \theta_1)$$

$$\theta_0 = temp\theta_0$$
$$\theta_1 = temp\theta_1$$

$$J = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i)^2$$

$$= \frac{1}{2m} \sum_{i=1}^{m} ((\theta_0 + \theta_1 x_1^i) - (y^i))^2$$

$$= \frac{1}{2m} \sum_{i=1}^{m} ((\theta_0 + \theta_1 x_1^i)^2 + (y^i)^2 - 2y^i(\theta_0 + \theta_1 x_1^i))$$

$$= \frac{1}{2m} \sum_{i=1}^{m} ((\theta_0)^2 + (\theta_1 x_1^i)^2 + 2\theta_0 \theta_1 x_1^i + (y^i)^2 - 2y^i\theta_0 - 2y^i\theta_1 x_1^i)$$

$$= \frac{1}{\cancel{2}m} \sum_{i=1}^{m} (\cancel{2}\theta_0 + \cancel{2}\theta_1 x_1^{(i)} - \cancel{2}y^i)$$

$$= \frac{1}{m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x_1^i - y^i)$$

$$\boxed{\frac{\partial J}{\partial \theta_0} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y^i)}$$

↑ Gradient respect to θ_0.

$$\frac{\partial J}{\partial \theta_1} = \frac{1}{m} \sum_{i=1}^{m} (\theta_0 x_1^i + \theta_1 (x_1^i)^2 - x_1^i y^i)$$

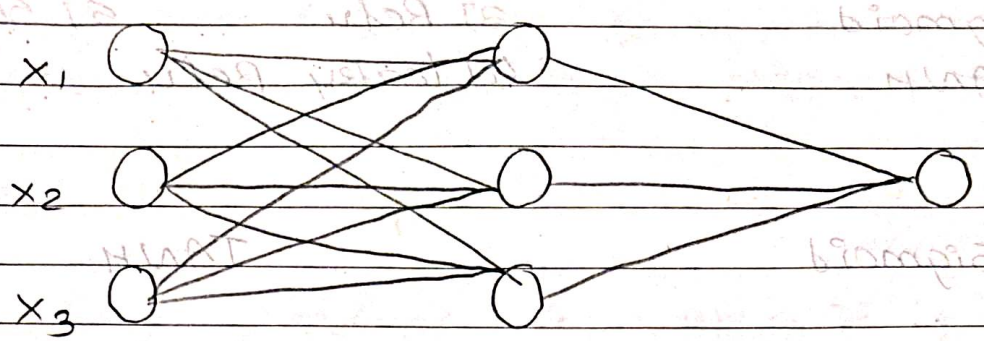
↑ Gradient respect to θ_1

$$= \frac{1}{m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x_1^i - y^i)\, x_1^{(i)}$$

$$= \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i)\, x_1^{(i)}$$

$h_\theta(x) = \theta$

Activation Function:



Input layer Hidden layer Output layer

Input = x_1, x_2, x_3

Function $y = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$

where w_1, w_2, w_3 are weights., b = bias.

- Weights:
  They tell the neural network how important are the input features for the given output.

- Now the equation is linear always. So for non linear we can't apply this.

- Activation is introduced to bring non-linearity into neural network.

- Cross - Entropy :-

$$D(\hat{y}, y) = -\frac{1}{N} \cdot \sum y_i \cdot \log(\hat{y}_i)$$

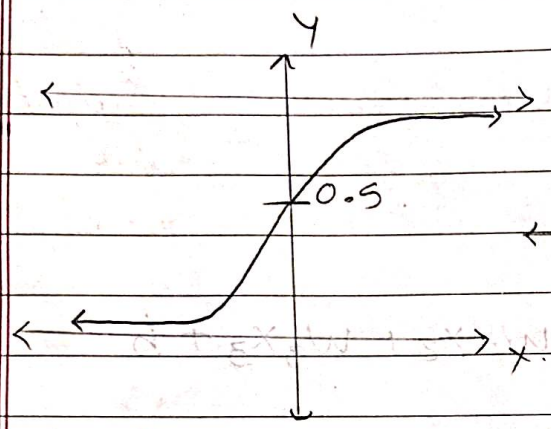
Note: y must be one hot encoded & $\hat{y}$ = softmax.

$\hat{y}$ = y-predicted.
y = y-true

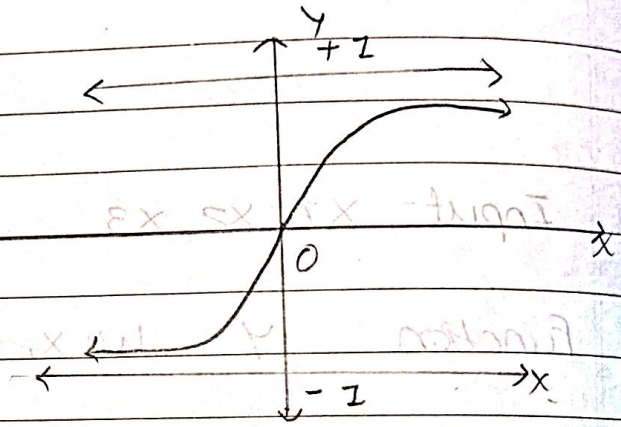
Types of Activation function.

1] Sigmoid
2] TANH
3] Relu
4] leaky Relu
5] Softmax

Sigmoid



TANH.

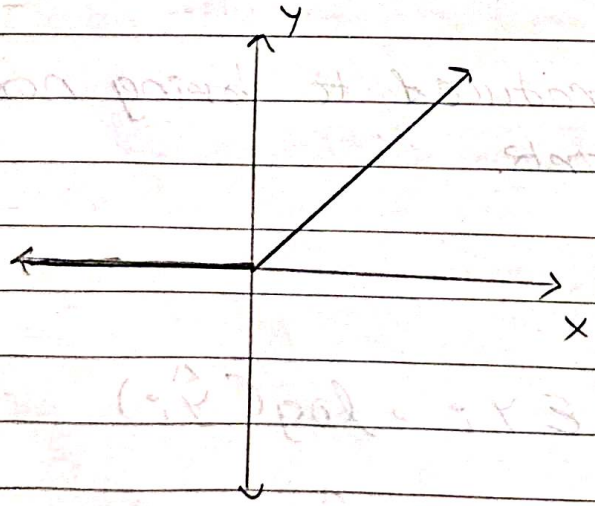


* → Problem: Vanishing Gradient.

$$\sigma(x) = \frac{1}{1+e^{-x}} \qquad \tanh x = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$

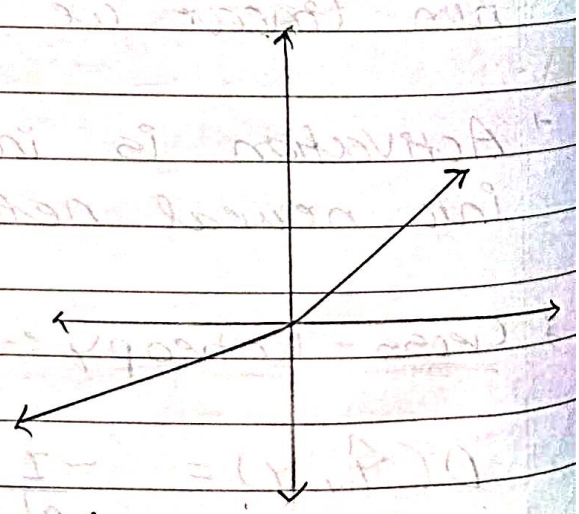
* → That's why we use relu in hidden layers

Relu



$Y = Max(0, x)$

Leaky Relu.



$Y = X, \; X>0$

$A*X, \; X<0$

↳ slope = 0.01 (defaut)

- Softmax is same as sigmoid. (Almost) $= S(x_i) = \frac{e^{x_i}}{\sum_{i=1}^{n} e^{x_i}}$
- It is used for multiple classes.
- Sum of output of all output neurons is 1
- For binary class we use sigmoid.

- Issues:

- Vanishing gradient in sigmoid & TANH

After some time the values for both graph gets constant so the activation function after that situation won't be able to find the loss, so it won't be able to adjust weights.

- That's why we use Relu in hidden layers.

- Issue in Relu is vanishing gradient for negative value to overcome that we use leaky Relu.

Relu (Rectified linear unit)

- It makes your network spars, problem "Dying Relu", It cause some nodes to completly die and not learn anything.

* Usages:

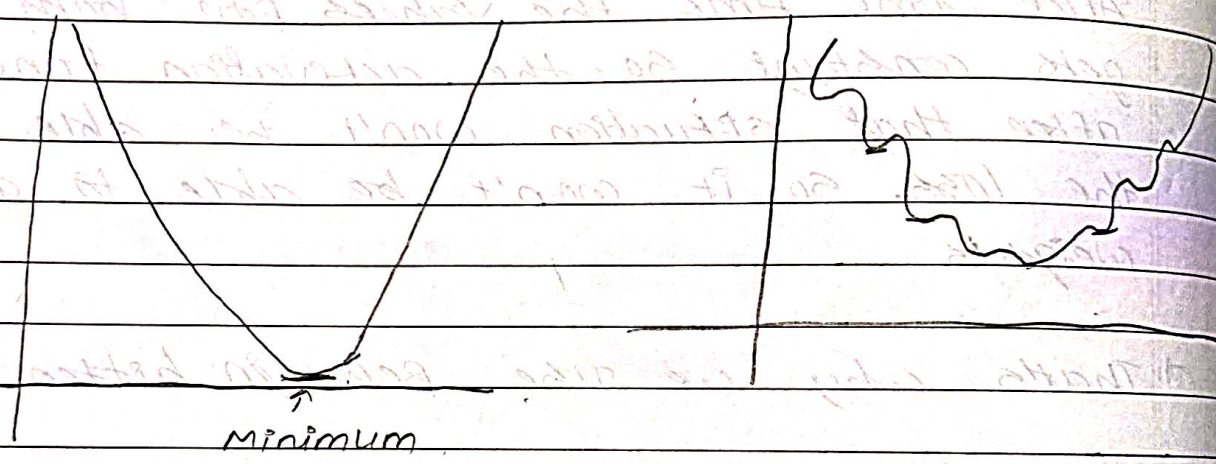
- Sigmoid is used when we want binary classificatio
- Softmax is used in Categorical / Multi class classification.

* Sigmoid :

$$\frac{1}{m} \sum_{i=1}^{m} \boxed{\frac{1}{2} \left(h_\theta(x^{i}) - (y^{(i)}) \right)^2}$$

cost $(h_\theta(x), y)$.

Convex function



→ If we apply mean squared error we will get local minima, so we have to find global minima so the cost will be minimum.

$$J(\theta) = \frac{-1}{m} \sum_{i=1}^{m} \left(y^{(i)} \log h_\theta(x^{i}) + (1-y^{(i)}) \log (1-h_\theta \right.$$

Cost function for logistic regression classification

$-\log h_\theta(x^{(i)})$ $y=1$

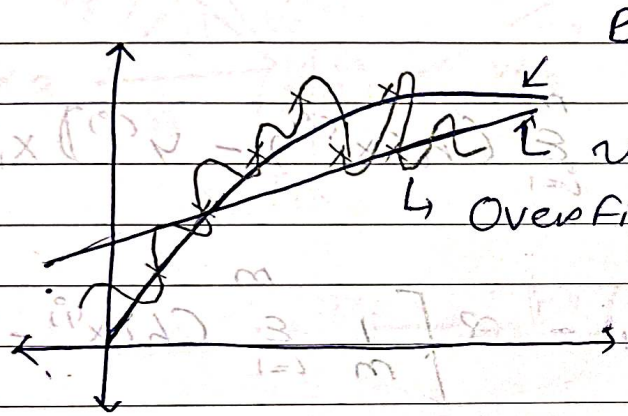
$-\log (1-h_\theta(x))$ $y=0$

$h_\theta(x) = g(\theta^T x)$

$$\theta_j = \theta_j - \alpha \frac{dJ}{d\theta_J}$$

$$\frac{dJ}{d\theta_J} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)}$$

* Fitting



↑ & low varience

under Fitting = high biased.

↳ Overfitting = high varience.

↳ & low biased.

1] Linear 2] Polynomial 3] Logistic Regression.

* Addressing Overfitting.

1] Collect data
2] Select features.
3] Regularization.

↳ $$J = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \sum_{j=1}^{n} \frac{\lambda}{2m} \theta_j^2$$

$$= \frac{1}{2m} \sum_{i=1}^{m} \left[(h_\theta(x^{(i)}) - (y^{(i)})^2) + \lambda \sum_{j=1}^{n} \theta_j^2 \right]$$

We don't regularize θ_0 because it is not associated with any features.

Now the Gradient decent will also change

$$\theta_j = \theta_j - \alpha \boxed{\frac{dJ}{d\theta_j}} \quad j = 1, \dots, n.$$

This will also change

For $j = 1$

$$\frac{dJ}{d\theta_1} = \frac{1}{m} \sum_{i=1}^{m} (h(x^{(i)}) - y^{(i)}) x_1^{(i)} + \frac{\lambda}{m}\theta_1$$

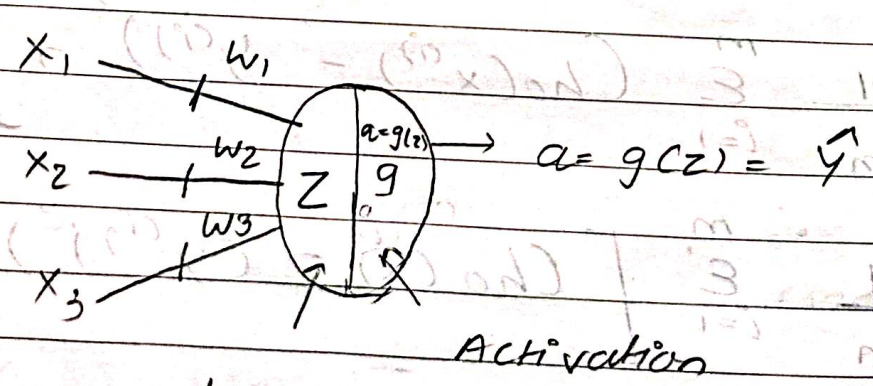
$$\theta_1 = \theta_1 - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} (h(x^{(i)}) - y^{(i)}) x_1^{(i)} + \frac{\lambda}{m}\theta_1 \right]$$

$$\theta_1 = \theta_1 - \frac{\alpha\lambda}{m}\theta_1 - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} (h(x^{(i)}) - y^{(i)}) x_1^{(i)} \right]$$

$$\theta_1 = \theta_1 \left(1 - \frac{\alpha\lambda}{m} \right) - \quad ''$$

So we have multiplied the θ_1 with something so the model will not be overfitted.

* Neural Network

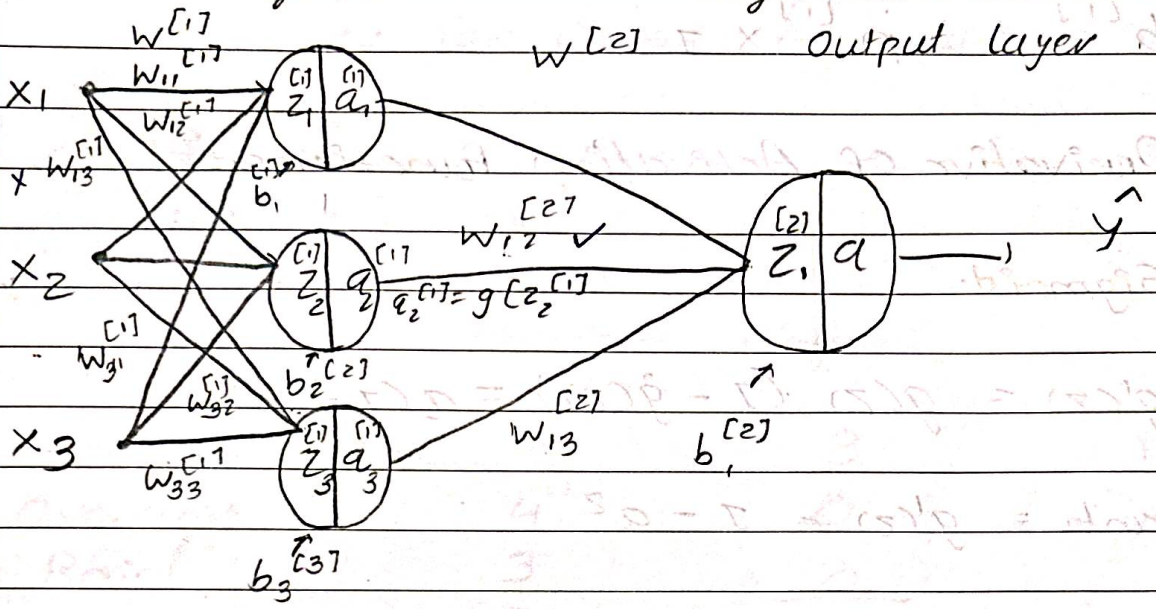


we can apply g as sigmoid, relu,

$W_{j\,k}^{[i]}$ to layer-i

to neuron j From neuron k

$$\boxed{z = w^T x + b} \longrightarrow \boxed{a = g(z)}$$



$$\boxed{z^{[1]} = w^{[1]} x\, b^{[1]}} \longrightarrow \boxed{a^{[1]} = g(z^{[1]})} \longrightarrow \boxed{z^{[2]} = W^{[2]} a^{[1]} + b^{[2]}}$$

$$\boxed{\hat{y} \leftarrow a^{[2]} = g(z^{[2]})}$$

$$z_1^{[1]} = W_{11}^{[1]} x_1 + W_{12}^{[1]} x_2 + W_{13}^{[1]} x_3 + b_1^{[1]} = z_1^{[1]}$$
$$W_{21}^{[1]} x_1 + W_{22}^{[1]} x_2 + W_{23}^{[1]} x_3 + b_2^{[1]} = z_2^{[1]}$$
$$W_{31}^{[1]} x_1 + W_{32}^{[1]} x_2 + W_{33}^{[1]} x_3 + b_3^{[1]}$$

$$z_1^{[1]} = \begin{bmatrix} W_{11}^{[1]}, & W_{12}^{[1]}, & W_{13}^{[1]} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$z^{[1]} = \begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \end{bmatrix} \qquad z_1^{[1]} = (W_1^{[1]})^T + b_1^{[1]}$$

$$W^{[i]} = n^{[i]} \times n^{[i-1]}$$

i = layer

$n \times n$ = Matrix size.

* Recurrent Neural Network :

Types of RNN:

1] One to One =>

2] One to many => Image Captioning

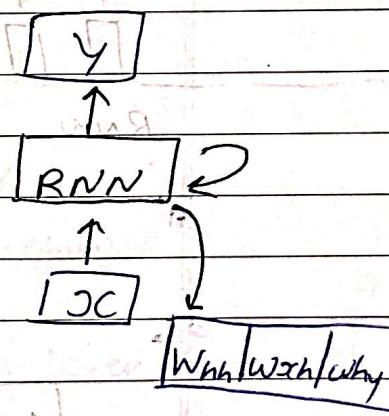
3] many to One => Sentiment Classification

4] many to many => Machine ~~Learning~~ Translation

Function:

$$h_t = F_w(h_{t-1}, x_t)$$

New State — Some function with para. w — Old State — Input vector at Some time t



$$h_t = \tanh(\underbrace{\underset{128\times128}{W_{hh}}\ \underset{128\times1}{h_{t-1}}}_{128\times1} + \underbrace{\underset{128\times256}{W_{xh}}\ \underset{256\times1}{x_t}}_{128\times1})$$

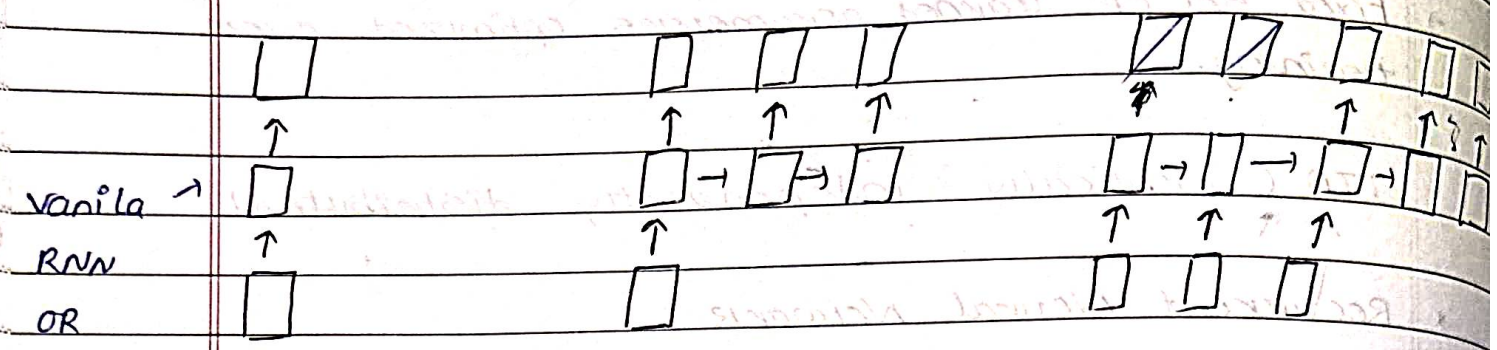
$$\underset{32\times1}{y_t} = \underbrace{\underset{32\times128}{W_{hy}}\ \underset{128\times1}{h_t}}_{32\times1}$$

Whh — Output — Input

Wxh — Output — Input

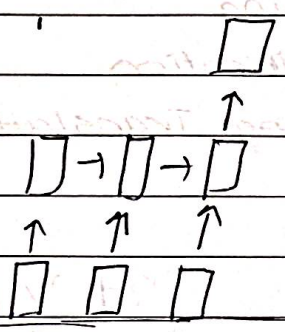
Whh → Fully Connected Network → h_{t-1} → Whh → ~~Whh~~ → $W_{hh} * h_{t-1}$

One to One: One to Many Many to Many



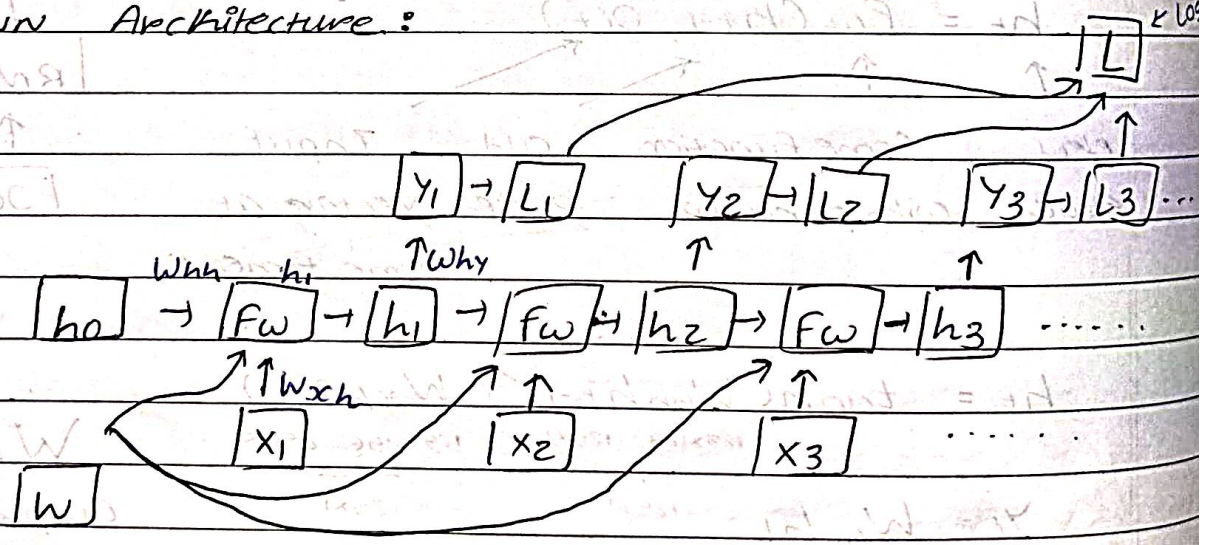
Video Captioning
Sequence of video Frames

Many to One.



min-char-rnn.py gist
github = karpathy.

RNN Architecture:

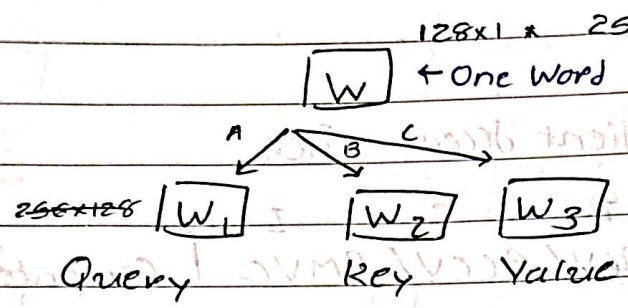


Many to Many

- Combine Many to One and One to Many
  - Many to One → Encoder
  - One to Many → Decoder

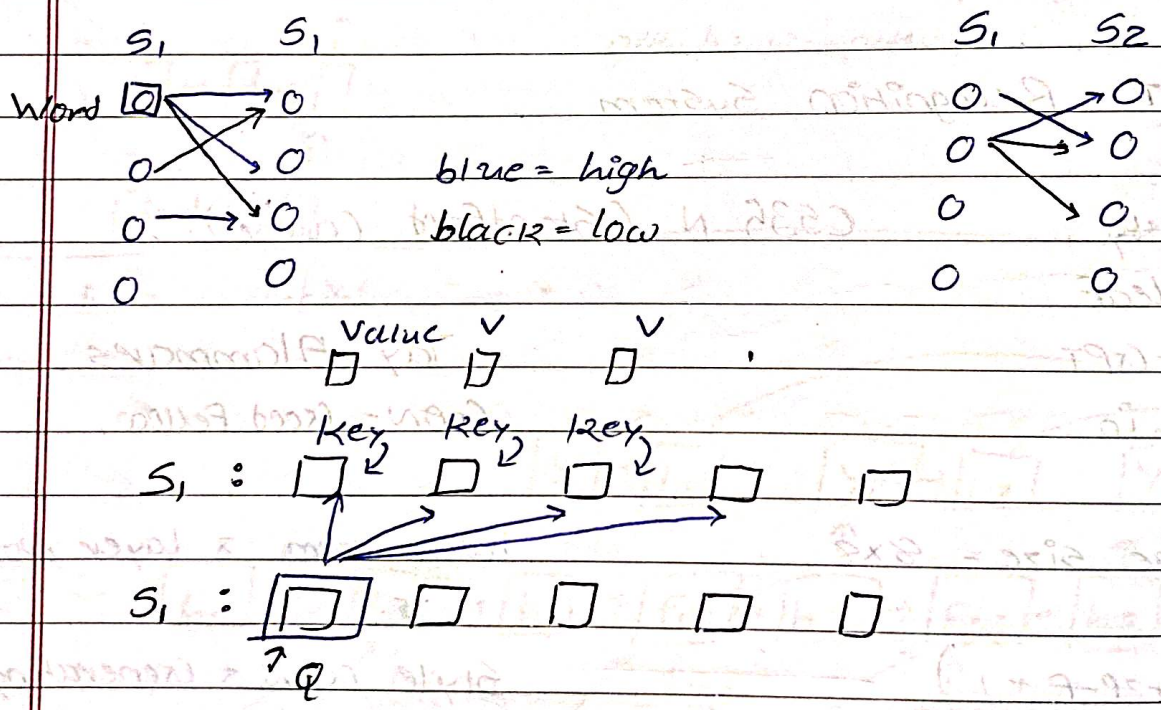
We use layer-normalization in this. we can't use batch-normalization.

* Transformers:



Attention:

Self



It will give a query & find the distance with all the words. Find the distance with cosine between 2 vectors.

← Attention Vector

Encoder Part

W_1 , W_2 , W_3 , W_4 , W_5

W_1^{QKV} W_2^{QKV}

$\forall_j \; W_i^{Q} * W_j^{K} = \boxed{W_1^{q_1}}\boxed{W_2^{q_2}}\boxed{W_3^{q_3}}\boxed{W_4^{q_4}}$ SoftMax — Small $\underline{a}$

$= \text{SoftMax}\left(\frac{Q \cdot K^T}{\sqrt{d_K}}\right)$

$\mathcal{E} \; W_i^{a_j} * W_j^{V} = \boxed{\text{////}}$ W_1 ← New Representation of old word

$Z = \text{SoftMax}\left(\frac{Q \cdot K^T}{\sqrt{d_K}}\right) \cdot V$

New Representation

Je vais bien

Representation

$\boxed{\text{Encoder}} \longrightarrow \boxed{\text{Decoder}}$

Masked Self Attention.

I am good

We need cross attention for this

Representation

Encoder-2

Feed-Forward

Multi-head attention

Encoder-1

Feed-forward Network

Multi-head attention

Embeded Values that comes from Word 2 Vec

I	- - - -	x_1
am	- - -	x_2
good	- - -	x_3

W^Q ⊗ (input Matrix) W^K W^V

* Multi-head Attention:

$W_1 \rightarrow H_1$

$Q|K|V \rightarrow W_1 N_1$

H_2

$Q|K|V \rightarrow W_1 N_2$

H_3

$Q|K|V \rightarrow W_1 N_3$

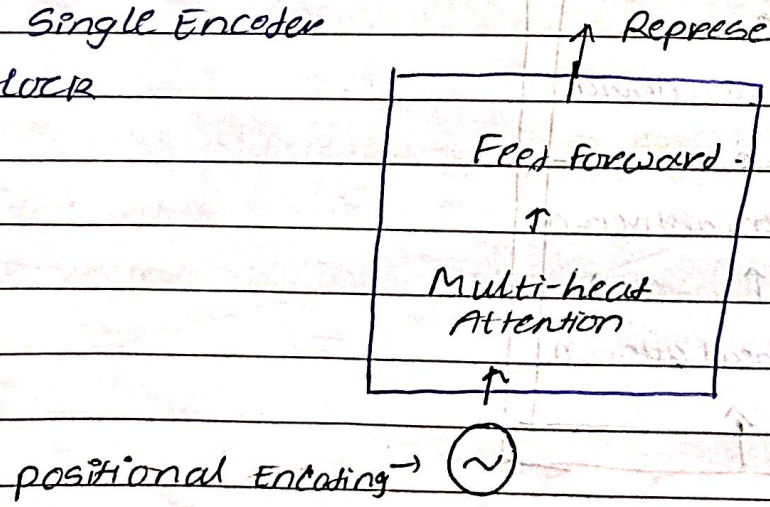
Concate → Feed Forward Network

Example = Apple
Fruit
Company

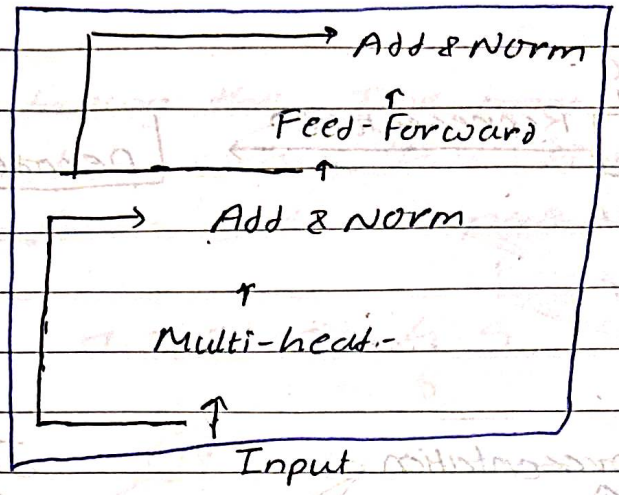
Multi-head attention = Concatenate $(Z_1, Z_2, Z_3, \ldots, Z_8)\, W_0$

Network

* A Single Encoder Block

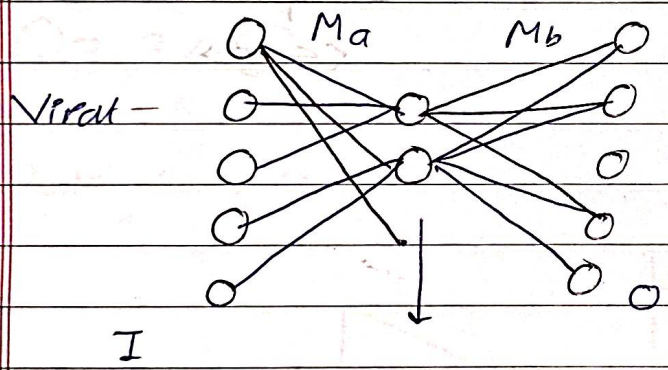


* Encoder block with the add & Norm component



* Word 2 Vec

Virat -



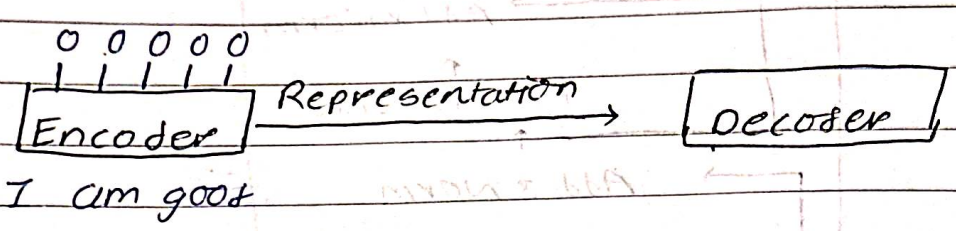
$M_a \times I = h$ &
$M_b \times h_2 =$ Out.

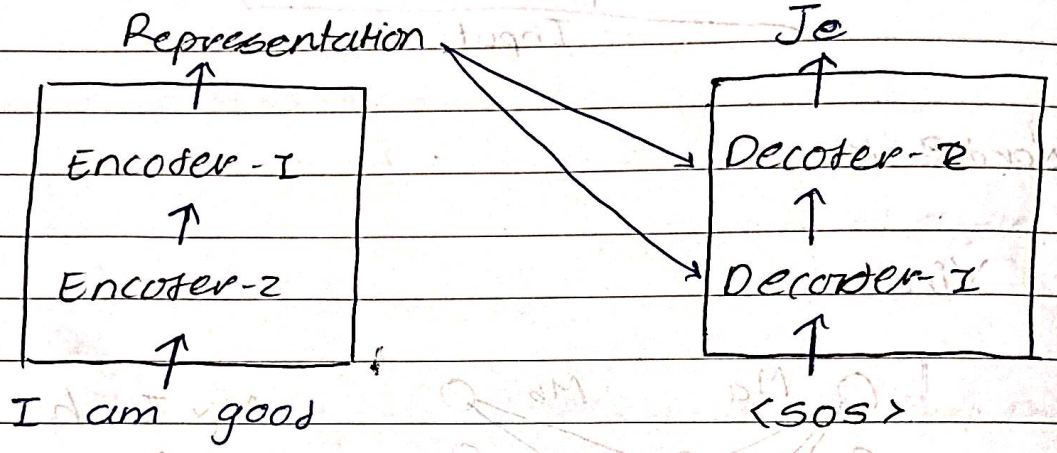
Vector / Hidden Representation.

Word = 5×1 (5-d I hot encoded)

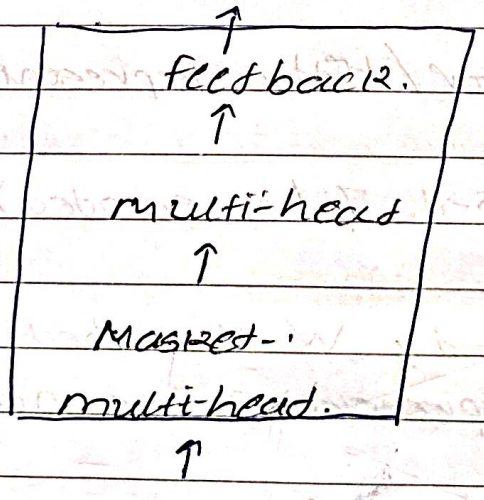
Vec - 2×1 (2-d Word embedding)
(preserving the notion of similarity)

* Decoder :





* Decoder block

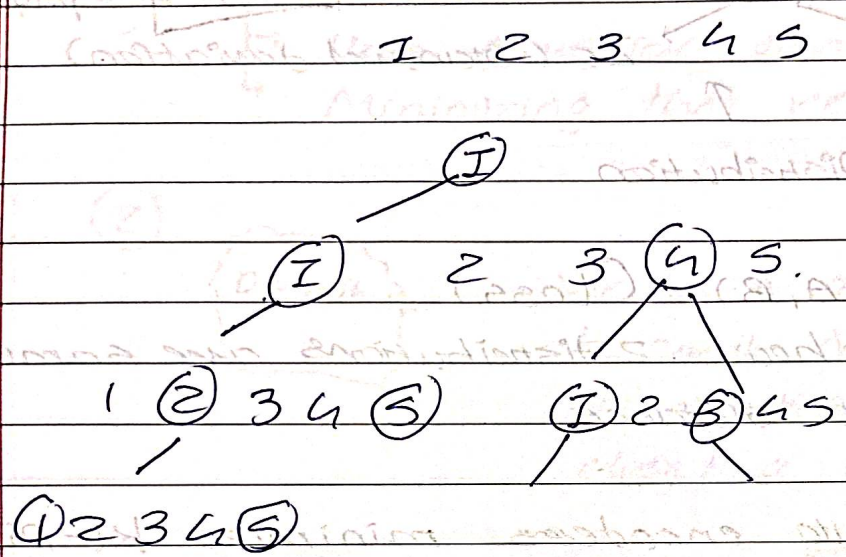


* Combination of encoder & decoder = Transformer
* Series of only encoder is BERT (Bi-directional, Encoder Representation of Transformers)
* Series of only decoder is GPT (Generalization Pre-trained)
  ↳ LLM

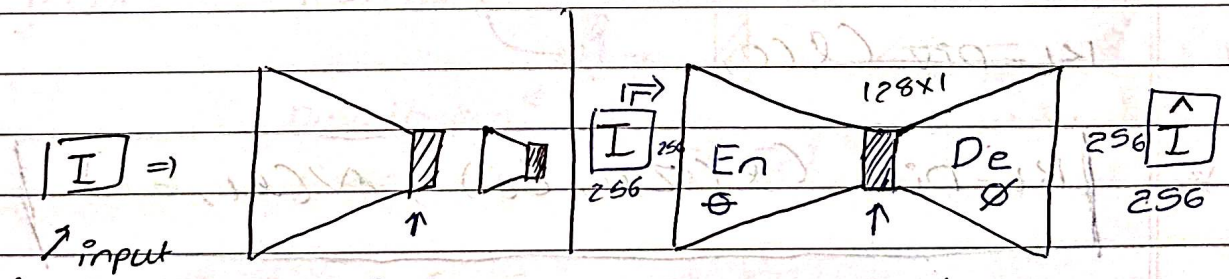
Date
Page

Beam - Search:

→ Instead of taking the one most probability word. we choose two words that has most probability and find output from both the paths.



* Auto- Encoder:



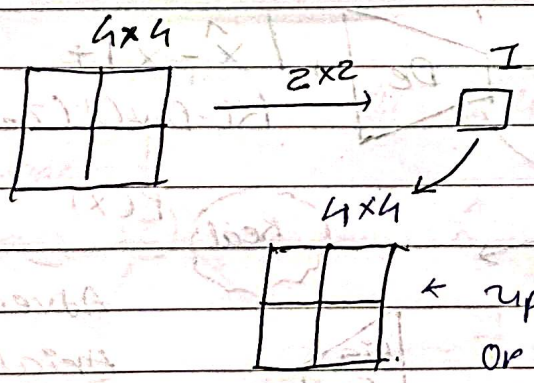
$I \xrightarrow{\text{Noise}} I^N$

denoising

BottleNeck.
or Latent space

Latent space

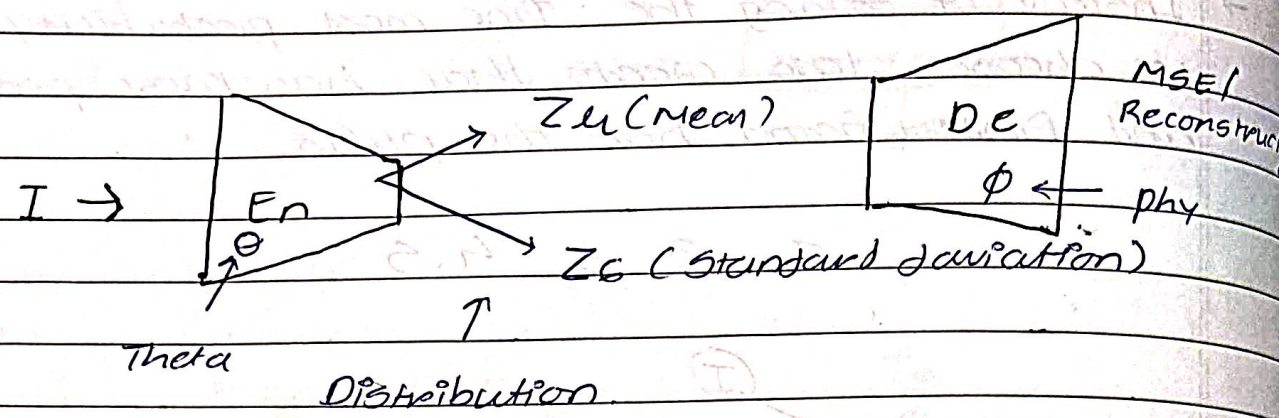
$loss = |\hat{I} - I|$



← up sampling.
or deconv

VAE:

* To generate image ~~for~~ we need to learn it's normal / gaussian distribution



* KL-Diversion (A, B) (Loss)
→ It will give how 2 distributions are same by finding the distance

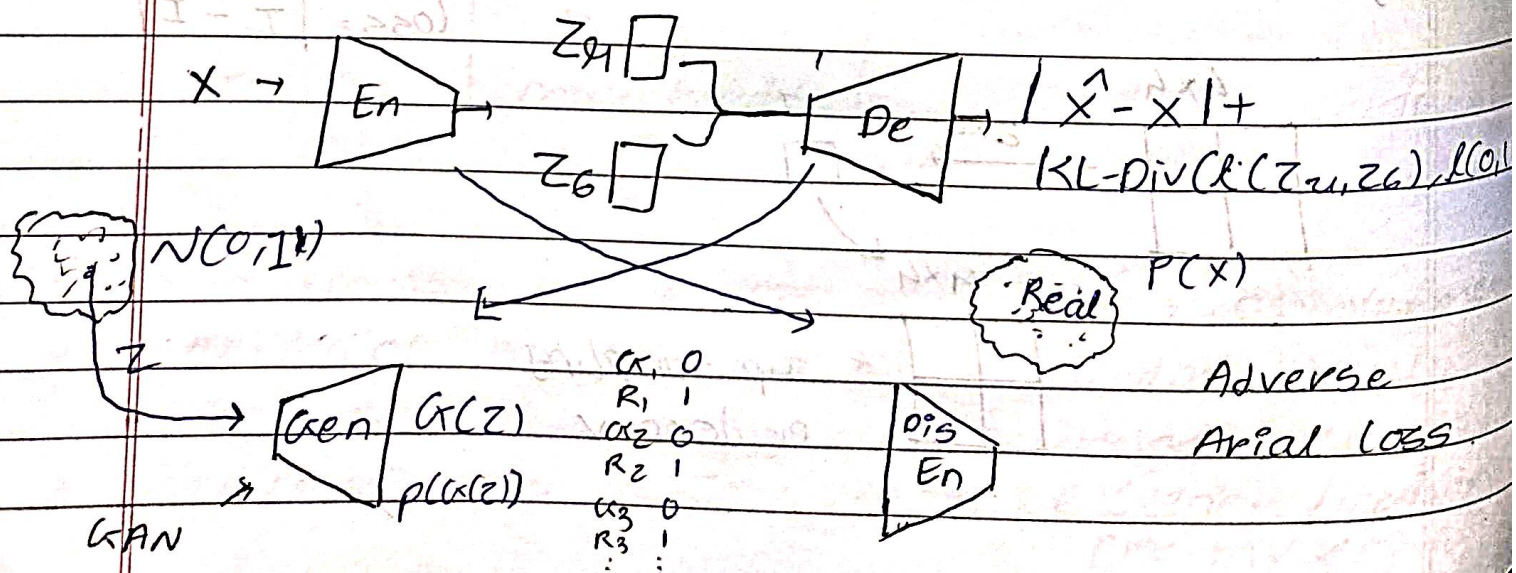
→ Variation auto encoders minimize KL-Div

→ GAN minimize Gension sennon distribuli

~~KL-Div (l(0,~~

* $\boxed{\text{KL-Div}\ (l(Z_\mu, Z_\sigma), N(\mu, \sigma))}$

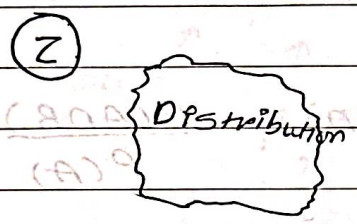
GAN:



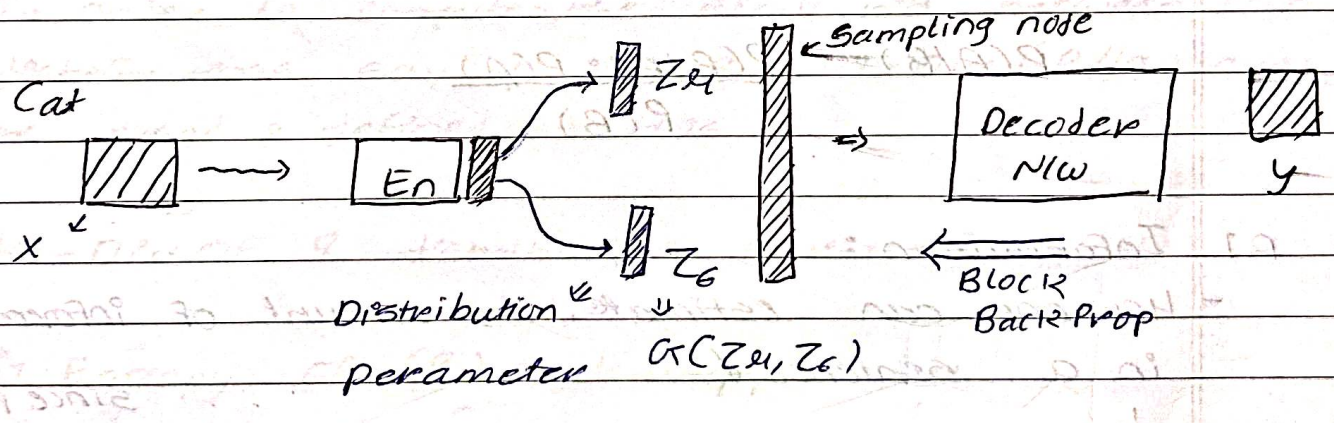
Stable diffusion:

Variational Auto Encoder:

→ In VAE we constraint our encoder network to generate latent vector (Z) that roughly follow
- Unit gaussian distribution
- Minimizing the reconstruction error.

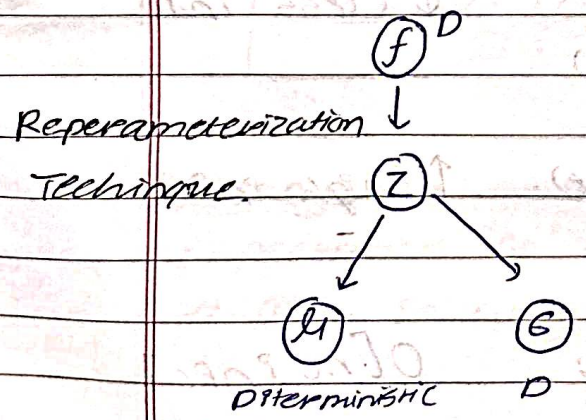


We force Z so that it must be comb. coming from a distribution which is very close to a Unit gaussian distribution



a] Re L : MSE (Y − X)

b] KL-D : KL-Div ($\sigma(Z_\mu, Z_\sigma)$, N(0,1))



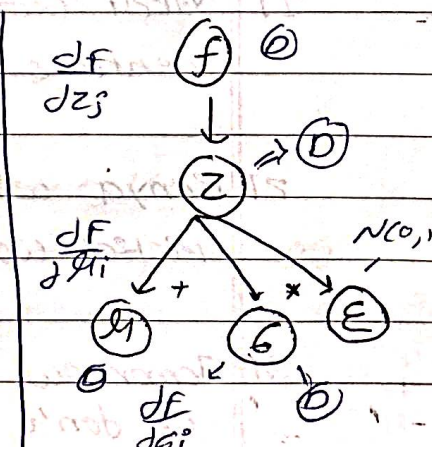
$\mu = Z_\mu$, $\sigma = Z_\sigma$

$\Downarrow$

$\hat{Z} \in N(0,1)$

$\Downarrow$

$\hat{Z} = Z_\mu + Z \cdot Z_\sigma$



$$KL\text{-}Div = (G(z_\mu, z_\sigma), N(0,1))$$

$$= \frac{1}{2} \sum_{i=1}^{N} \left(\mu_i^2 + \sigma_i^2 - \log(1e-8 + \sigma^2) \right) - 1$$

A] * Bayes Theorem:

Conditional probability are A & B

$$P(A|B) = \frac{P(A \cap B)}{P(B)}, \quad P(B|A) = \frac{P(A \cap B)}{P(A)}$$

Hypothisis. Evidence

$$\Rightarrow P(A|B) \cdot P(B) = P(B|A) \cdot P(A)$$

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

B] Information:

→ How one can estimate the amount of information in a sentence?

	X (Event)	P(x) (Prob)	$-\log(P(x))$ (info)
			since $p(x) \in (0,1)$ hence $(-\log)$
1]	Virat scored a centure	↑ (highly probable event)	↓ (less info
2]	Kenya wins cricket worldcup	↓ (rare event)	↑ (high info)
3]	Tomorrow it rain or don't	1 (centered event)	0 [no info]

⇒ So basically rare events carry more information

C] Average of Information is Entropy (H):

→ The expected value of info with respect to any event (x) averaged over all value (X) can attain is Entropy (H)

$$H = -\Sigma\, p(x) \log(p(x))$$ ← Info content in any x

Sum of all x

prob x

D] KL-Divergenes (KL-Div)

→ In order to compute the similarity between two distributions say p and q, KL-Div (p||q) can be used & defined as this.

KL-Div of q distribution w.r.t P,

i.e. Entropy of (q) - Entropy of (P)

↓ Ammount of info in (q) distribution

↓ Ammount of info in (P) distribution.

$$\boxed{-\Sigma q(x) \log(q(x)) + \Sigma p(x) \log(p(x))}$$

KL-Div is almost this except that the expectation is always computed w.r.t p(x) as KL-Div is w.r.t p(x).

ii) $\boxed{-\mathcal{E}\, p(x) \log q(x) + \mathcal{E}\, p(x) \log p(x)}$

This is crossentropy between p and q distributions

This is entropy of (p)

KL-Div is the diff between average info of q(x) wrt p(x) that of p(x) w.r.t p(x)

$$\text{KL-Div}(p(x)/q(x)) = -\mathcal{E}\, p(x) \log q(x) + \mathcal{E}\, p(x) \log p(x)$$

$$= \mathcal{E}\, p(x) \log \frac{p(x)}{q(x)}$$

$$= -\mathcal{E}\, p(x) \log \frac{q(x)}{p(x)}$$

* KL-Div is not symmentic as
KL-Div $(p/q) \neq$ KL-Div (q/p)

Hence it is a distance measure ~~between a~~ b/w divergence.

* KL-Div ≥ 0 It is always +ve and b/w (0 & 1)

We want to find the prob. of x given θ (Param)

ELBO (Evidence lower bound)

⇒ Suppose we have given 1 million images of 100x100 pixels.

→ We are assuming this pixels by pixel image sampling experiment as a stochastic process and can be modeled using random variables.

$\boxed{M_1}$ $P(x) = P(x_1, x_2, x_3, \dots x_{10000})$ Joint prob distri

$P_\theta(x)$ where

$P_\theta(x)$ = Probability of $(x \in T)$ with the distribution parameterized over $[\theta]$ $T = \theta$: Training dataset.

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \, [P(x \in T)]$$

We know that pixel are not independent in an image so we have to find the joint probability

$\boxed{M_1}$ $P(x) = P(x_1, x_2 \dots x_{10000}) = P(x_1) \cdot P(x_2/x_1) \cdot P(x_3/x_1, x_2) \dots$

Computation is impossible

* So There is huge ammount of dependency between random variable (x's), we can assume another set of hidden/latent variables say $(z_1, z_2, \dots z_{100})$ that can influence $P(x)$ directly.

Basically our image (x) get influenced by few factors (z) such as pose, noise...

Since dimensions of z is fewer than x it is easy to get hold of $p_\theta(x)$ via z.

$\boxed{M_2}$ $$p(x) = \int_z p(x, z) \cdot dz = \int_z p(x|z) \cdot p(z) \cdot dz$$

(Marginalized over all z's)

Hense it is marginalized estimation is also not feasible.

M_1 & M_2 ~~& M3~~ are not possible so try something else.

$\boxed{M_3}$ $$p(z|x) = \frac{p(z, x)}{p(x)}, \quad p(x) = \int_z p(z, x) \cdot dz.$$

Instead of statistically estimating $p(z|x)$ we wanted to approximet

$p(z|x)$ via $q(z) \in \mathcal{L}$ ⇒ Family of some known tractable distribution. (Ex unit normal, Gaussian)

$$q^*(z) = \text{argmin } KL\text{-Div}(q(z) / p(z|x))$$

Date
Page

$$\text{KL-Div}(q(z) / P(z|x)) = -\mathop{E}_{z}\, q(z) \log \frac{P(z|x)}{q(z)}$$

We wanted to maximize $P_\theta(x)$

$$\text{KL-Div}(q(z) / P(z|x)) = -\mathop{E}_{z}\, q(z) \log \frac{p(z|x)}{q(z)}$$

$$= -\mathop{E}_{z}\, q(z) \log \frac{p(x,z)}{p(x) \cdot q(z)}$$

$$* \quad = -\mathop{E}_{z}\, q(z) \left[\log \frac{p(x,z)}{q(z)} \times \right.$$

$$= -\mathop{E}_{z}\, q(z) \left[\log \frac{p(x,z)}{q(z)} - \log p(x) \right]$$

$$= -\mathop{E}_{z}\, q(z) \log \frac{p(x,z)}{q(z)} + \log p(x) \underbrace{\mathop{E}_{z}\, q(z)}_{1}$$

$$= -\mathop{E}_{z}\, q(z) \log \frac{p(x,z)}{q(z)} + \log p(x)$$

$$* \quad \log p(x) = \text{KL-Div}(q(z) / P(z|x)) + \mathop{E}_{z}\, q(z) \log \frac{P(x,z)}{q(z)}$$

Constant Always

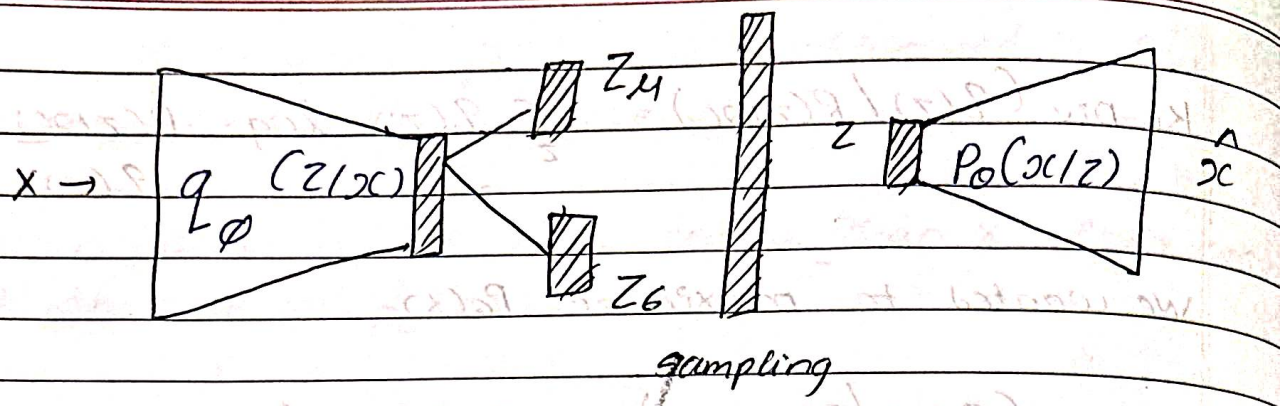
KL-Div ≥ 0

Minimize (B)

Variational lower bound at $P(x)$ is the evidence this will be called as Elbo: As a Result this is Maximized (C)

Constant (A)

Elbo: Evidence lower bound.



ELBO:

$$L(\theta, \phi; x, z) = \underbrace{E_{q_\phi(z|x)} \log P_\theta(x|z)}_{\text{Reconstruction}} - \underbrace{D_{KL}(q_\phi(z)/p(z))}_{\text{KL-Div}}$$

This Needed to Maximized. (A)

Maximized. (B)

Minimized. (C)

B is the log likelihood of x given z, Maximizing it ensures z should have to generate x. (Decoder)

D = but these z's are coming from Encoder hence averaged over $q_\phi(z|x)$ (Encoder)

C = Minimizing this KL-Div. makes $q(z)$ as close to $p(z) = N(0,1)$

$$L = \mathop{E}_{z} q(z) \log \frac{p(x,z)}{q(z)} \qquad (z \sim q_\phi(z|x))$$

Evidence lower bound

$$= E\, q(z) \log \frac{P(x|z) * P(z)}{q(z)}$$

$$= \underbrace{\mathcal{E}\, q(z) \log p(x|z)}_{A} + \underbrace{\mathcal{E}\, q(z) \log \frac{p(z)}{q(z)}}_{B}$$

$$\boxed{\mathbb{E} \log P_\theta(x/z) - \text{KL-Div}\,(q(z)/P(z))}$$

$q_\phi(z|x)$ ↳ Encoder

↳ Decoder

Final VAE loss:

$$L(\phi, \theta; x, z) = \min \|x - \hat{x}\| + KL(q(z)/p(z))$$

* GAN (Generative Adversarial Network)

Here in VAE we used MSE as loss.

We know that

$$L = \frac{1}{2 \cdot M \cdot N} \sum_{i}^{M} \sum_{j}^{N} (\hat{x}_{ij} - x_{ij})^2 \quad \text{(Basic MSE/L2 loss)}$$

$$L \propto \frac{1}{2} \|\hat{x} - x\|$$

* Look at the Gaussian distribution

$$\underbrace{P(x/\mu, \sigma^2)}_{} = \frac{1}{Z} e^{\frac{(-\mu - x)^2}{2\sigma^2}}$$

Means

For the given μ, σ

Normalization Constant

With what prob this x belongs to this distribution or not

Let us assume $\mu = \hat{x}$ and $\sigma^2 = 1$

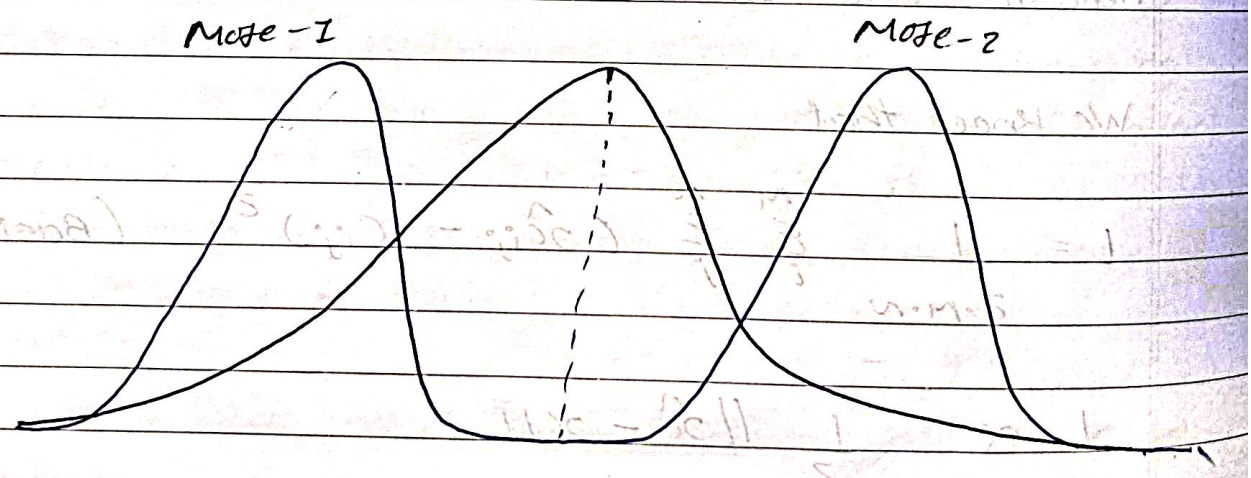
$$P(x/\hat{x}) \propto e^{-\frac{1}{2}\|\hat{x}-x\|_2^2}$$

Taking log both side

$$\log P(x/\hat{x}) \propto \frac{1}{2}\|\hat{x}-x\|_2^2$$

Meaning L2/MSE minimization means log likelihood Maximization

7] Problem with MSE



- If we have 2 class dog & cat then with MSE loss it will always gives the average answer
- Average answer means blurred images.

21 Problem with KL-Div.

KL-Div
Forward
$KL(P||Q)$
$= \mathcal{E}\, p(x) \log \frac{p(x)}{Q(x)}$

KL-Div
Reverse.
$KL(Q||P) =$
$\mathcal{E}\, Q(x) \log \frac{Q(x)}{P(x)}$

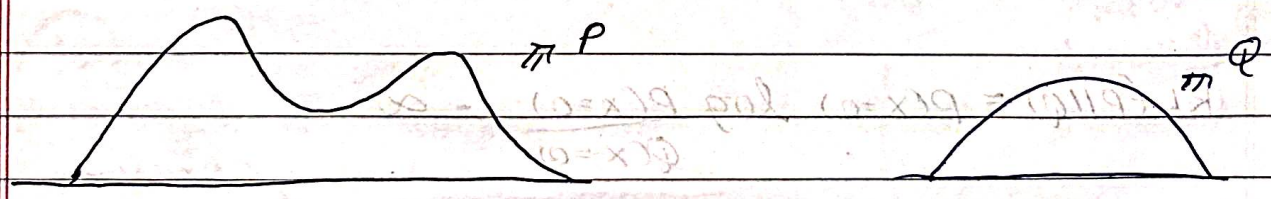
* Why we use ~~Forward~~ Reverse KL-Div in VAE?

Estimating p via Q = Forward

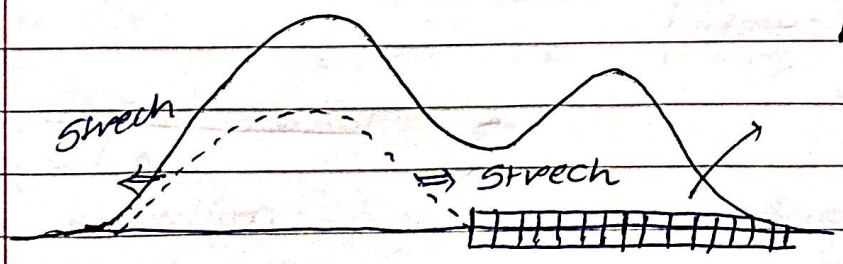
$\min_{\phi} \{ KL(P||Q_{\phi}) \}$

$\min_{\phi} \left(\mathcal{E}_{x}\, p(x) \log \frac{P(x)}{Q(x)} \right)$ What happens when $Q(x)=0$ and $p(x) \neq 0$
F-KL → ∞ / not defined.

Bimodel (P) (unknown) unimodel (assumed).



Now



Mean seeking

Here $Q(x)=0$ but $p(x) \neq 0$
F → KL → ∞.

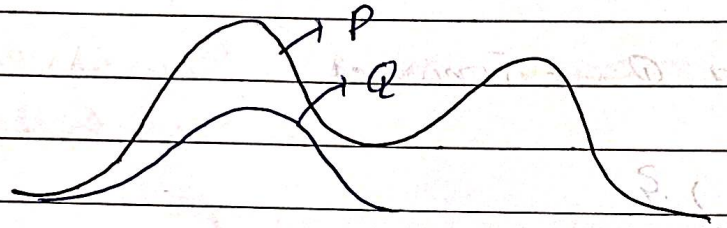
Over estimation

* Reverse KL-DN

$$\min_{\phi} \left\{ \sum_{x} Q_{\phi}(x) \log \frac{Q_{\phi}(x)}{P(x)} \right.$$

$Q(x) \neq 0$
$P(x) = 0$.
Blow it up.

→ Mode seaking



* → Here also problem occurres, here Q will only learn to generate only 1 type of image either dog or cat.

So we have to find some new loss.

$$KL(P||Q) = P(X=0) \log \frac{P(X=0)}{Q(X=0)} = \infty$$

$$KL(Q||P) = Q(X=0.5) \log \frac{Q(X=0.5)}{P(X=0.5)} = \infty$$

No overlapping

New loss:

JS-Divergense:

$$JSD(P||Q) = \frac{1}{2}\left[KL(P||M) + KL(Q||M)\right]$$

where $M = \frac{P+Q}{2}$

We use JSD loss in GAN

GAN:

- The sequence of Decoder (Generator) and Encoder (Discriminator) get changed. Their objective are also tweaked keeping the overall objective of image reconstruction being fixed. Instead of unexplicitly learning the image distribution (as in VAE) GAN's explicitly learns it.

GAN in Action

Date
Page

Loss Function: Binary Cross entropy for classification.

$$L(\hat{y}, y) = y \log \hat{y} + (1-y) \log (1-\hat{y})$$

Discriminator loss function:

I/P ⇒ Either $\underline{\underline{x}}$ OR $\underline{\underline{G(z)}}$

$\hat{y}$ O/P (label) ⇒ $D(x)$ OR $D(G(z))$

y Ground Truth (label) ⇒ 1 (Real) OR 0 (Fake)

* $L(D(G(z)), 0) = \log [1 - D(G(z))]$

here D need to minimize $D(G(z))$ ≡ Maximize $1 - D(G(z))$

⇓

Maximize $\log (1 - D(G(z)))$

* $L(D(x), 1) = \log(D(x))$

Here D need to minimize $D(x)$

$D(x)$ maximization $\equiv \log(D(x))$ Maximization

Final Discriminator Function:

$$\max_{D} \left[\log(D(x)) + \log(1 - D(G(z))) \right]$$

		P	N	Predicted.
Actual	P	TP	FN	
	N	FP	TN	

here we are increasing the probability of this.

Over a batch:

* $$\max_{D} \left[\mathbb{E}_{x \sim p(x)} (\log(D(x))) + \mathbb{E}_{z \sim P_z(z)} (\log(1 - D(G(z)))) \right]$$

data

Discriminator loss function

* Generator needs to fool Discriminator by generating images as real as possible hence:

$$\min_{G} \mathbb{E}_{z \sim P_z(z)} (1 - D(G(z)))$$

Since G is never going to see any real data, just for completness one can write generations complete loss function as.

$$\min_{G} \left[\underset{x \sim p(x)_{data}}{E} [\log(D(x))] + \underset{z \sim p_z(z)}{E} [\log(1 - D(G(z)))] \right]$$

* Total Combined loss Function:

$$\min_{G} \max_{D} V(D, G) =$$

$$\min_{G} \max_{D} \left(\underbrace{\underset{x \sim p(x)_{data}}{E}}_{\text{Real data}} [\log(D(x))] + \underbrace{\underset{z \sim p_z(z)}{E}}_{\text{Random Vector}} [\log[1 - D(\underbrace{G(z)}_{\text{Image generated by Random Vector}})]] \right)$$

* Point - 1

→ First we will show that given any fixed (G) the optimally achieved D^*_G can be defined as.

$$D^*_G(x) = \underset{D}{\text{Arg Max}} \; V(D, G)$$

$D^*_G(x)$ → Best possible D; $V(D, G)$ → Fixed G

$$= \frac{P_{data}(x)}{P_{data}(x) + P_g(x)}$$

Date
Page

* Point - 2

→ Secondly for any given (say best $D^*_G(x)$) The optimal G^* can be defined as:

$$G^* = \underset{G}{\text{Arg min}} \; V(D^*_G, G)$$

↑ Fixed ↳ To get best generator.

→ It is been shown that this optimization have a unique solution achieved when

$\boxed{P_g = P_{data}}$ desirable from the problem pt self

→ Also it has been shown that it is equivalent to minimize $JSD(P_{data} \| P_g)$.

* Proving Point - 1 :

→ For a given G our objective is to get the optimal D

$$D^*_G = \underset{D}{\text{Arg min}} \; V(D, G)$$

↓ best possible ↳ For a given G

$$D^*_G = \underset{D}{\text{Arg Min}} \left\{ \mathbb{E}_{x \sim p_{data}(x)} [\log(D(x))] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))] \right.$$

$$\underset{P(x)}{\mathbb{E}}[x] = \int_{x} x \, P_x(x) \cdot dx$$

Expected Value of $R.V(x)$, Following PDF $\rightarrow P(x)$ Random Variable.

Hence $V(G, D)$:

$$V(G,D) = \int_{x} \overbrace{P_{data}(x) \cdot \log(D(x)) \cdot dx}^{A}$$

$$+ \int_{z} \overbrace{P_z(z) \cdot \log(1 - D(G(z)))}^{B} \cdot dz$$

* Random Variable Transformations:

$$\underset{\substack{P_x(x) \\ \text{Given}}}{X} \rightarrow \boxed{\underset{\text{Transformer}}{G}} \rightarrow \begin{array}{l} Y = G(X) \\ P_Y(Y) = (?) \end{array}$$

$$P_Y(y) = P_x(G^{-1}(y)) \cdot \frac{d}{dy}(G^{-1}(y)) \quad - (1)$$

Assuming $X = G^{-1}(Y)$

Now consider case of GAN's.

$$\underset{P_z(z)}{Z} \rightarrow \boxed{G} \rightarrow X = G(z) \Rightarrow z = G^{-1}(X)$$

$P_g(x)$ or $P_x(x)$

using eq-I

$$P_g(x) = P_z(G^{-1}(x)) \cdot \frac{d}{dx}(G^{-1}(x)) \quad - \text{(ii)}$$

* Term b in equation

$$\int_z P_z(z) \log(1 - D(G(z))) \cdot dz$$

Changing of variable let $z = G^{-1}(x)$, $x = G(z)$

$$dz = \frac{d(G^{-1}(x))}{dx} \cdot dx$$

$$\int_x \underbrace{P_z(G^{-1}(x)) \cdot \log(1 - D(x)) \cdot \frac{d}{dx}(G^{-1}(x))}_{P_g(x) \quad \text{(using ii)}} \cdot dx$$

$$\int_x P_g(x) \cdot \log(1 - D(x)) \cdot dx$$

* Final $V(G, D)$:

$$\int_x P_{data}(x) \cdot \log(D(x)) \cdot dx + \int_x P_g(x) \cdot \log(1 - D(x)) \cdot dx.$$

$$D^*_G(x) = \underset{\text{Max } D}{\text{Arg}} \; [V(D, G)]$$

↳ optimal → Fixed.

Hence in order to maximize $V(D,G)$ we need to differentiate it w.r.t $[D(x)]$.

$$\frac{d(V(D,G))}{d[D(x)]} = \frac{P_{data}(x)}{D(x)} - \frac{P_g(x)}{1 - D(x)} = 0$$

$$\boxed{D^*_G(x) = \frac{P_{data}(x)}{P_{data}(x) + P_g(x)}}$$

→ Practically can't see computed but analytically proved.

* Now for generator.

$$G^* = \underset{G}{\text{Arg min}} \; V(D^*_G, G)$$

$$G^* = \underset{G}{\text{Arg min}} \left\{ \int_X P_{data}(x) \cdot \log(D^*_G(x)) \cdot dx + \int_X P_g(x) \cdot \log(1 - D^*_G(x)) \cdot dx \right.$$

$$= \underset{G}{\text{Arg min}} \left\{ \int_X \left(P_{data}(x) \cdot \log\left(\frac{P_{data}(x)}{P_{data}(x) + P_g(x)} \right) + \right.\right.$$

$$\left.\left. P_g(x) \cdot \log\left(\frac{P_g(x)}{P_{data}(x) + P_g(x)} \right) \right) \cdot dx \right\}$$

$$\left.\begin{array}{l} + \log(2) \cdot P_{data}(x) - \log(2) \cdot P_{data}(x) \\ + \log(2) \cdot P_g(x) - \log(2) \cdot P_g(x) \end{array}\right\} \text{Extra term added.}$$

Combine them.

$$= \underset{G}{\text{Arg min}} \left\{ \int_X \left(-\log(2) \left(P_{data}(x) + P_g(x) \right) \right.\right.$$

$$+ P_{data}(x) \cdot \log \frac{2 \cdot P_{data}(x)}{P_{data}(x) + P_g(x)}$$

$$\left.\left. + P_g(x) \cdot \log \frac{2 \cdot P_g(x)}{P_{data}(x) + P_g(x)} \right) \cdot dx \right.$$

Hence:

$$G^* = \underset{G}{\text{Arg min}} \left[-\log(4) + \int_X P_{data}(x) \cdot \log\left(\frac{2 \cdot P_{data}(x)}{P_{data}(x) + P_g(x)} \right) \cdot dx \right.$$

$$\left. + \int_X P_g(x) \cdot \log\left(\frac{2 \cdot P_g(x)}{P_{data}(x) + P_g(x)} \right) \cdot dx \right.$$

$$= \underset{G}{\operatorname{Arg\,Min}} \left[-\log(4) + 2 \cdot JSD\left(P_{data}(x), P_g(x)\right) \right]$$

This implies that Optimizing G^* is equivalent to minimizing JSD between $P_{data}(x)$, $P_g(x)$.

After the above minimization, convergence,

$JSD \to 0$, $P_g(x) \to P_{data}(x)$

and

$V(D, G) \to -\log(4)$

This shows that:

→ The global minima of the criterion fn $V(D_G^*, G)$ is acheived iff ($P_g = P_{data}$) and the loss will be equal to $[-\log(4)]$.

The corresponding loss can be $D_G^* = \dfrac{P_{data}}{P_{data} + P_g} = \dfrac{1}{2}$.