100 days ML by Campus X

Batch vs Online ML

Batch	Online
→ Only stored data is trained → hardware limitation and has lots of data	Real time - New data is trained. → Performance improved dynamically as the new data is added.

Tensors

Containers for storing numbers

Scalar → 0D Tensor

Vector → 1D Tensor

Cube → 2D Tensor

matrix → 2D Tensor

no of axis = Rank. Dimension

nD tensor =

One hot Encoding

Categorical Data

Ordinal

Nominal

↓

one hot encoding

If there are many categories in the data how do convert that to numbers.

[1,0,0] yellow
[0,1,0] blue
[0,0,1] Red
⋮

multicolinearity :-
Cant let input vectors to have multiplicative relation

n categories ——— (n-1) columns.

columns.

pd.get_dummies (data, column)

One hot encoder class in Scikit learn → Used for ML

ohe = OneHotEncoder ( )
ohe.fit_transform (x_train ["fuel", "owner"])

Column Transform

Nominal Categ. → City, Gender (ohe)

Age → Simple encoder

Ordinal Categ. → Review

[ ] [ ] [ ]

[

Column Transform packed all these arrays

Used to Categorize data and avoid many lines of code.

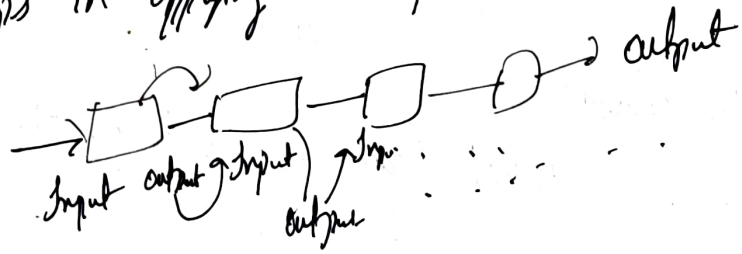
day 29

Pipelines

from sklearn

→ Chain together multiple steps so that output of each step can be used as input of next step

→ Helps in applying same preprocessing to train and test



Scoring function →

Decision tree

Pass list of tuple.

each tuple has list of transformers.

make_pipeline → tuple is easier

make_train

Binning and Binarization

Converting Numerical data → Categorical Data

Binarization

Discretisation (Binning)

- Unsupervised
  - → Equal width
  - → Equal frequency
  - → Kmeans binning
- Supervised
  - Decision tree binning
  - Rare
- Custom
  -

Equal width binning / Uniform

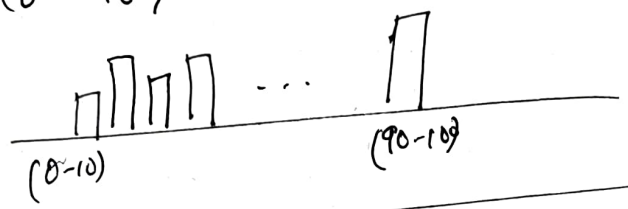
Bins = any number

let Bins = 10

$$\frac{max - min}{bins} = \frac{100 - 0}{10} = 20$$

Range =

(0 - 10), (10 - 20), (20 - 30) .... (90 - 100)] 10 bins



Equal Freq / Quantile Binning

Interval = any number

ex: 0 - 10 %. 10 - 20 %. ... 90 - 100 %.

← 10 %. 20 %. 100 %.

Intervals → helps Uniform Value Spread
→ widely used than Equal width

K-means binning

→ Used Clustering algorithm. Forms clusters automatically
Same algo is used.
→ Used when clusters can be formed in the data
→ here, Intervals = "Centroid".
→ line that bisects clusters
→ every point distance is measured from centroid
Center to centroid, assigned to that cluster.

Day 15

Data Gathering

- CSV
- JSON/SQL
- Fetch API
- Webscraping

CSV

Comma Seperated Value

" Pandas read csv . html " → webpage documentation

dtypes parameter → overriding data types parameter in the data file.

To convert float to int or anything like this

Dates → by default its Strings → object = String

Should convert to date - type time (datetime 64)

Keras functional Model

Non linear models

concatenating features from different models (RNN, CNN ... etc)

RNN

Recurrent Neural Network

used for sequential data and widely used in ~~tsNLP~~

Type of Sequential model.

example of sequential data → Text / Passage
→ Time series data
→ Audio / Speech.
→ DNA Sequence.

Why not NN?

- because Input size can not be varied in NN.

this can be solved but there still exists this problem when the input size increase to thousands words, million words ---

Calculation will increase ---!

Need to do lot of Unnecessary calculation

Thats why a new architecture only to predict Sequential data, they created RNN by Wilhelm Lenz and Ernst Ising

Road map → Simple RNN - architecture
Back prop in RNN in time
Problems with RNN
LSTM
GRU
Types of RNN
Deep RNN
Bi directional RNN
~~Sequence~~ Sequence to Sequence model

Data in RNN :-

Format = (timestep, input_features)

Can represent a sentence in One hot encoded format

movie was good bad not

[1,0,0,0,0] [0,1,0,0,0] [0,0,1,0,0] [0,0,0,1,0] [0,0,0,0,1]

Sentiment = 0, 1
-ve +ve

Sentiment length = 5

Sentance :-

1. x_1 movie (x_{11}) was (x_{12}) good (x_{13}) — 1
2. x_2 movie (x_{21}) was (x_{22}) bad (x_{23}) — 0
3. x_3 movie (x_{31}) was (x_{32}) not (x_{33}) good — 0

Representation = (3, 5)

3 → timestep

5 → # of input features

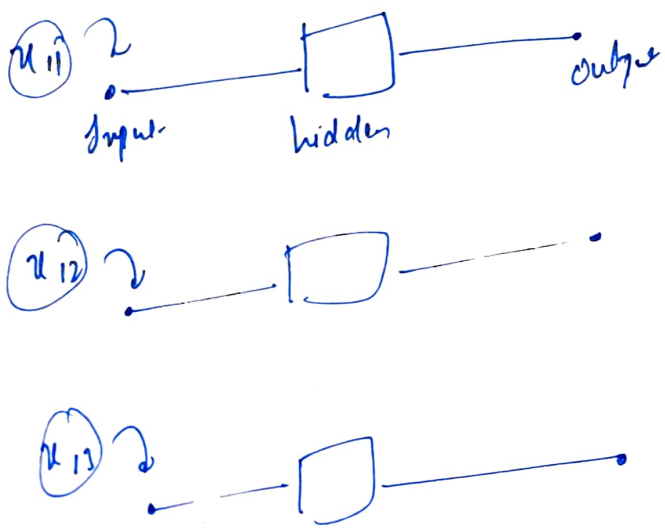
[0,0,1,0,0] → length = 5

In Keras → It has Simple RNN

format of input data = Batch_size, timestep, input_features

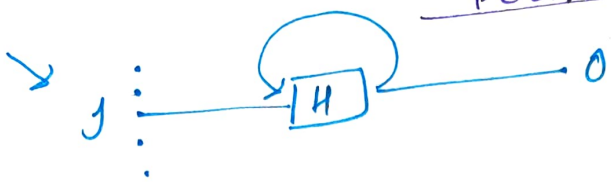
(3 → 4, 5)

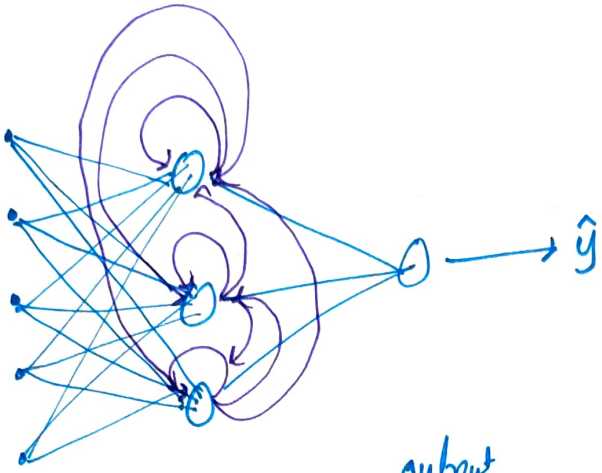
Structure :-



There is a concept of State in RNN

Feedback → Very Important

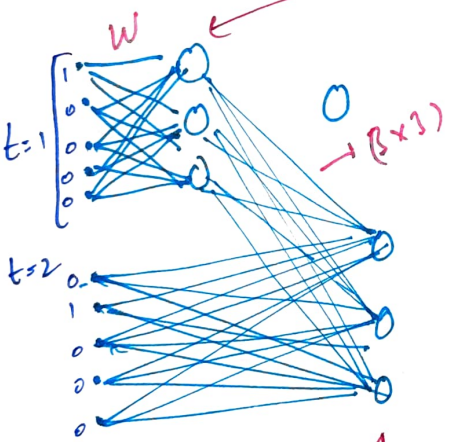




Input hidden output (Sigmoid)

Proper diagram :-

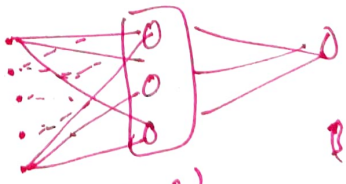
step 1 we send random values for feedback



#of weights and Biases? 27 + 4 Bias (weights)

W = 5 x 3 = (5, 3)

hidden = (3, 3)

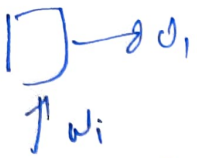


(5x3) (3x3) (3x1)

15 9 3 → 27 + 4 bias

Forward Pass → Unfolding through time

Bours in Recurrent loop



$t=1$ $x_{11} = [1,0,0,0,0]$

dot product with weights.

Inside Recurrent Network → Simple NN function

Silent Activation - tanh

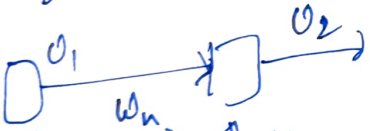
output

$O_1 = \tanh(x_{11} \cdot w_i)$ → tanh (n) relu

(1×3)

(1×3)

$t=2$ Same weights are used



t_1 x_{11} $x_{12} = [0,1,0,0,0]$

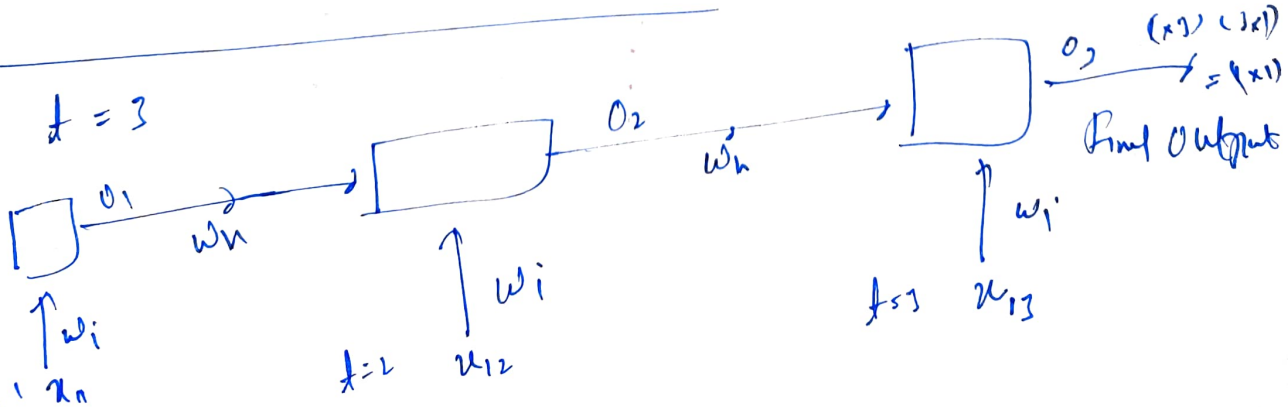
$O_2 = \tanh[x_{12} w_i + O_1 w_n]$

(1×5) (5×3) (1×1) (1×3)

(1×1)

(1×3)

$t=3$



$$O_j = \underbrace{\tanh\left(x_{i3}\, w_i + O_2\, w_h\right)}_{(1 \times 1)}$$

except from $t=1$, rest all time step is having 2 Inputs. [words and output's from other steps / previous

$t = 1$

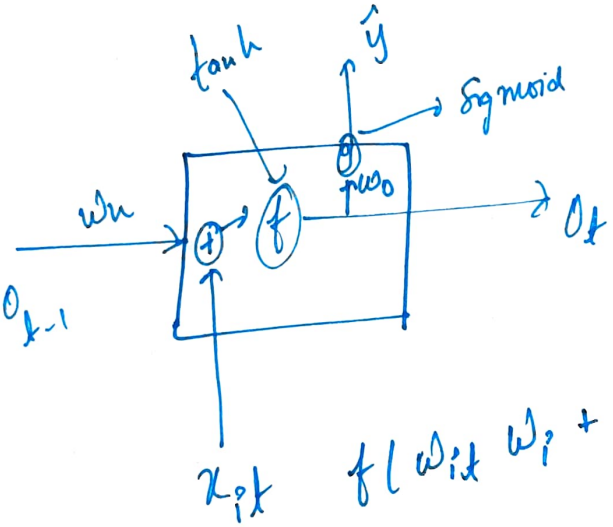
only 1 Inputs

So we give O_0 to keep consistancy with random number or not 0's

So $f_1 = \tanh\left(x_{11}\, w_i + O_0\, w_h\right)$ $[\because O_0 = 0]$

→ Parameter and weights sharing is used here

→ Simple RNN can process sequence of last 10 time steps.



$$\hat{y} = g\left(O_t\, w_0\right)$$

$$f\left(w_{it}\, w_i + O_{t-1}\, w_h\right) = O_t$$

i = row #

t : timestep

Embedding

does semantic meaning [words that are closer in vector space are expected to be closer in meaning]
reduces dimensions.
(by avoiding Padding)

Backpropagation through time [BPTT]

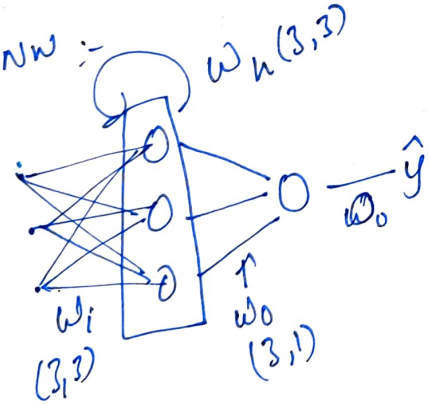
on Sentiment analysis model. , Many → One RNN

text → I/O

	x_0	x_{12}	x_{13}		
x_1	Cat	met	rat	1	[1,0,0] [0,1,0] [0,0,1]
	x_{21}	x_{22}	x_{23}		
x_2	rat	rat	met	1	[0,0,1] [0,0,1] [0,1,0]
	x_{31}	x_{32}	x_{33}		
x_3	met	met	Cat	0	[0,1,0] [1,0,0] [1,0,0]

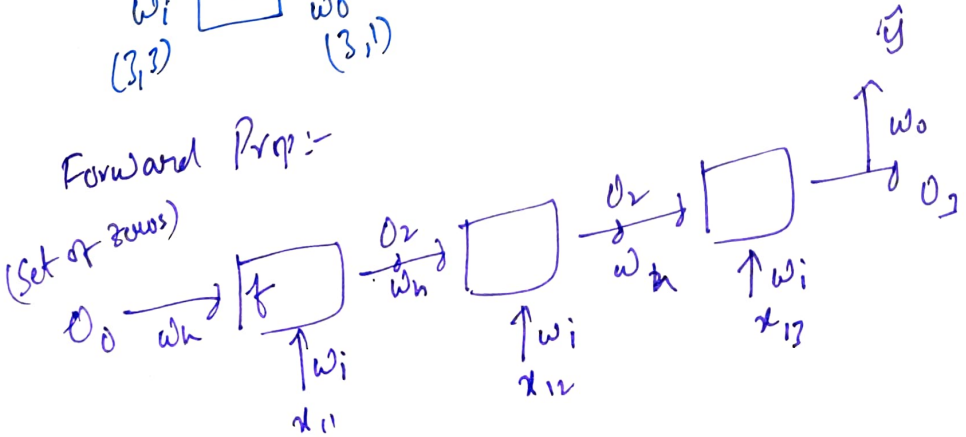
vocab = Cat [1,0,0] | mat [0,1,0] | Rat [0,0,1]

RNN :-



(3,3) } dimension
(3,1)
Ignoring bias here

Forward Prop :-



eq ① $O_1 = f(x_{11} w_i + O_0 w_h)$

eq ② $O_2 = f(x_{12} w_i + O_1 w_h)$

eq ③ $O_3 = f(x_{13} w_i + O_2 w_h)$

eq ④ $\hat{y} = \sigma[O_3 \cdot w_0]$

eq ⑤ $L = -y_i \log \hat{y}_i - (1-y_i) \log (1-\hat{y}_i)$ — loss fn

Forward Pass equations

Using Gradient descent to find backward pass :-

w_i w_h w_0

loss depends on $\hat{y}$, $\hat{y}$ depends on O_3, w_0. O_3 depends on previous term

$$w_i = w_i - \eta \frac{\partial L}{\partial w_i}$$

$$w_h = w_h - \eta \frac{\partial L}{\partial w_h}$$

$$w_0 = w_0 - \eta \frac{\partial L}{\partial w_0}$$

task is to find these 3 derivatives

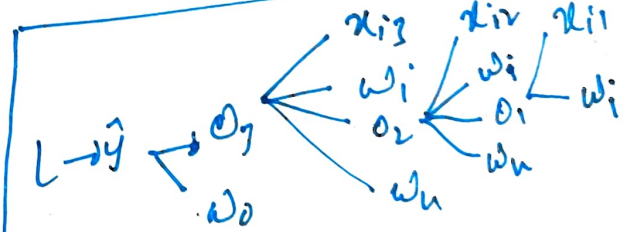
$$\frac{\partial L}{\partial w_0} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_0}$$

From eq ④ we can find $\frac{\partial \hat{y}}{\partial w_0}$

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial O_3} \cdot \frac{\partial O_3}{\partial w_i} +$$

$$+ \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial O_3} \cdot \frac{\partial O_3}{\partial O_2} \cdot \frac{\partial O_2}{\partial w_i}$$

$$+ \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial O_3} \cdot \frac{\partial O_3}{\partial O_2} \cdot \frac{\partial O_2}{\partial O_1} \cdot \frac{\partial O_1}{\partial w_i}$$



dependency chart of $L(w_i)$

this is because the RNN unfolded 3 times (because there are 3 words in input) if there are n words then there will be n equation

$$\frac{\partial L}{\partial w_i} = \sum_{j=1}^{3} \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial O_j} \cdot \frac{\partial O_j}{\partial w_i}$$

n = time stamps

$$\frac{\partial \hat{y}}{\partial O_1} = \frac{\partial \hat{y}}{\partial O_3} \cdot \frac{\partial O_3}{\partial O_2} \cdot \frac{\partial O_2}{\partial O_1}$$

$\rightarrow J=1$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial O_3} \frac{\partial O_3}{\partial O_2} \frac{\partial O_2}{\partial O_1} \frac{\partial O_1}{\partial w_i} \quad \Big\} \text{ for term 1}$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial O_3} \cdot \frac{\partial O_3}{\partial O_2} \cdot \frac{\partial O_2}{\partial w_i} \quad \Big\} \text{ for } J=2$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial O_3} \frac{\partial O_3}{\partial w_i} \quad \Big\} \text{ for } J=3$$

$$\frac{\partial L}{\partial w_i} = (J=1) + (J=2) + (J=3)$$

both are equal

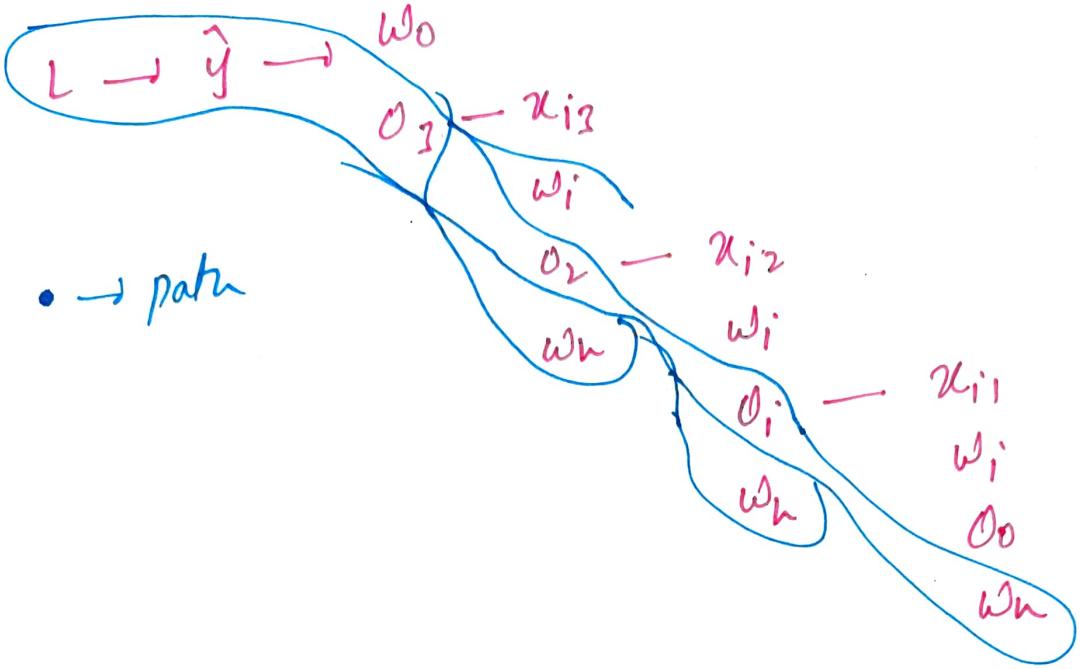
$$\frac{\partial L}{\partial w_h} = ?$$

$$\frac{\partial L}{\partial w_h} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial O_3} * \frac{\partial O_3}{\partial w_h} + \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial O_3} \frac{\partial O_3}{\partial O_2} \frac{\partial O_2}{\partial w_h} + \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial O_3} \frac{\partial O_3}{\partial O_2} \frac{\partial O_2}{\partial O_1} \frac{\partial O_1}{\partial w_h}$$

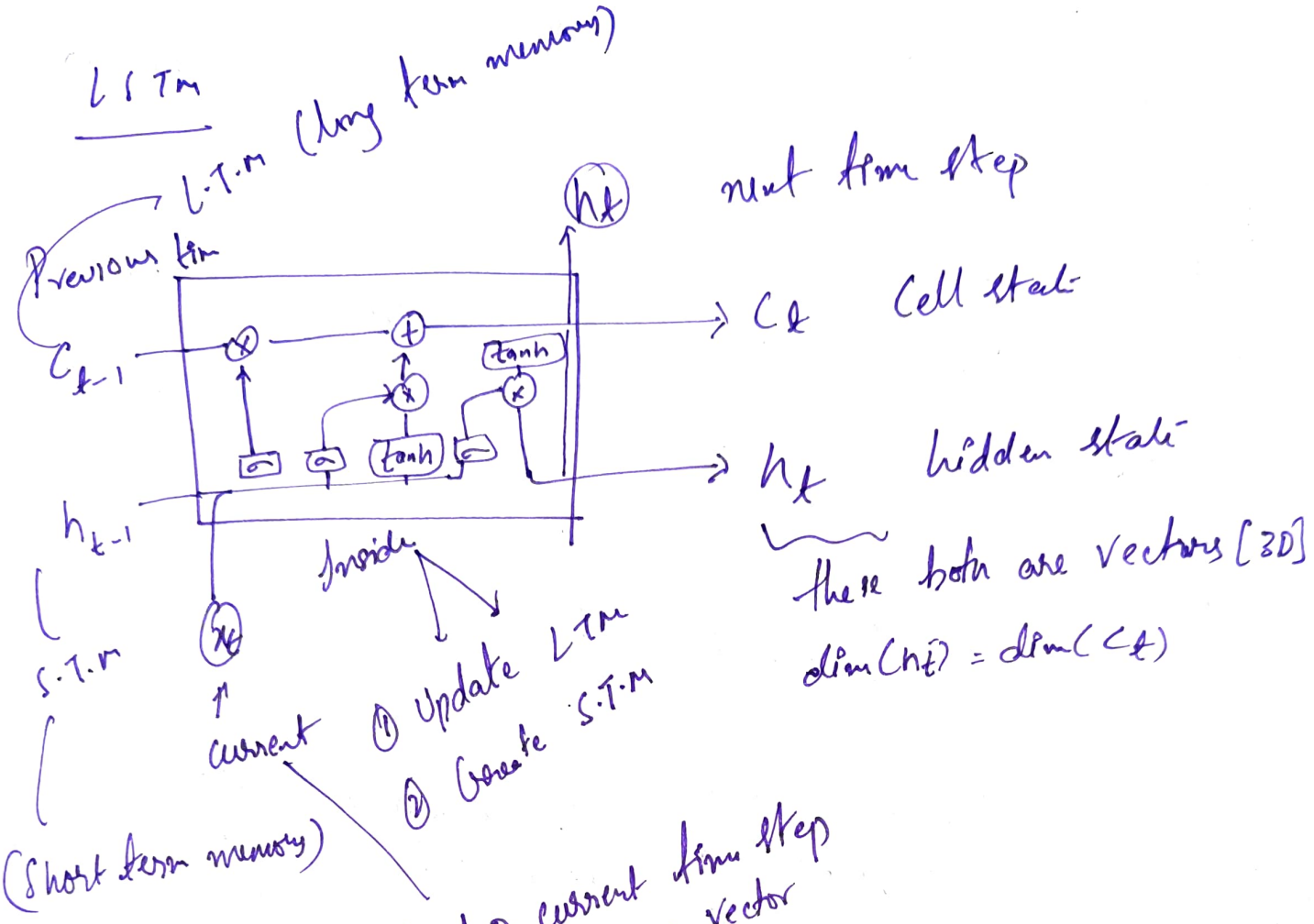
$$\frac{\partial L}{\partial w_h} = \sum_{j=1}^{3} \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial O_j} \cdot \frac{\partial O_j}{\partial w_h}$$

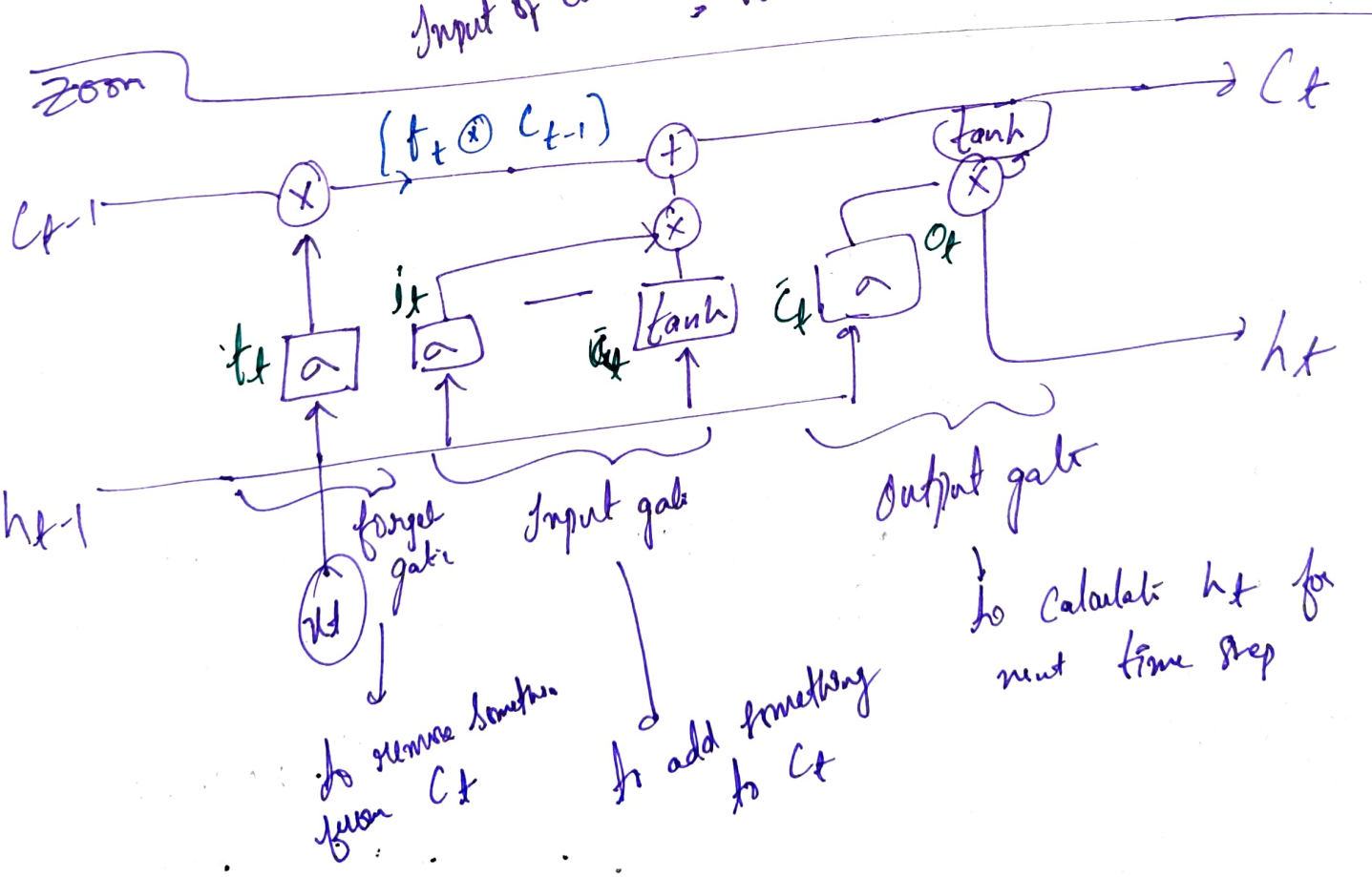
this is formed from the dependancy chart for w_h that is drawn in last paper. In next paper also

Let's do the dependency chart for w_n



LSTM


L.T.M (long term memory)
Previous time
next time step
C_{t-1}
C_t Cell state
h_t
h_{t-1}
h_t hidden state
these both are vectors [3D]
dim(h_t) = dim(C_t)
S.T.M
(short term memory)
x_t
current
Inside
① Update L.T.M
② Create S.T.M
Input of current time step
= vector
tanh
σ


Zoom
C_t
$(f_t \otimes C_{t-1})$
C_{t-1}
i_t
f_t
$\tilde{C}_t$
o_t
h_t
h_{t-1}
x_t
forget gate
Input gate
Output gate
to remove something from C_t
to add something to C_t
to calculate h_t for next time step
tanh
σ

the gates:-

② Input
① Forget
③ output

Forget Gate

multiplot encoded.

f_t is $\bar{C}_t$ → candidate cell state
forget input
O_t output } vector

$\dim(f_t) = \dim(\bar{C}_t) = \dim(f_t) = \dim(o_t)$

Pointwise operation = ⊕, ⊗, tanh
↳ output = vector

a tanh : neural networks
Sigmoid. collection of multiple nodes with sigmoid and tanh activations.

Forget Gate

C_{t-1} $f_t \otimes C_{t-1}$

f_t

h_{t-1}

x_t

Input = x_t : current stat
h_{t-1} : previous hidden stat

output : Can remove something from t_2 cell state.

Stages:
① Calculate f_t
② $C_{t-1} \otimes f_t$

let there be 3 units of NN

$f_t = [f_1\ f_2\ f_3]$

$(3\times 7)\ (7\times 1) + (3\times 1) = (3\times 1)$

→ $w_{3\times 7}$ all weight of this

h_{t-1} x_t (7×1) Dimen inp

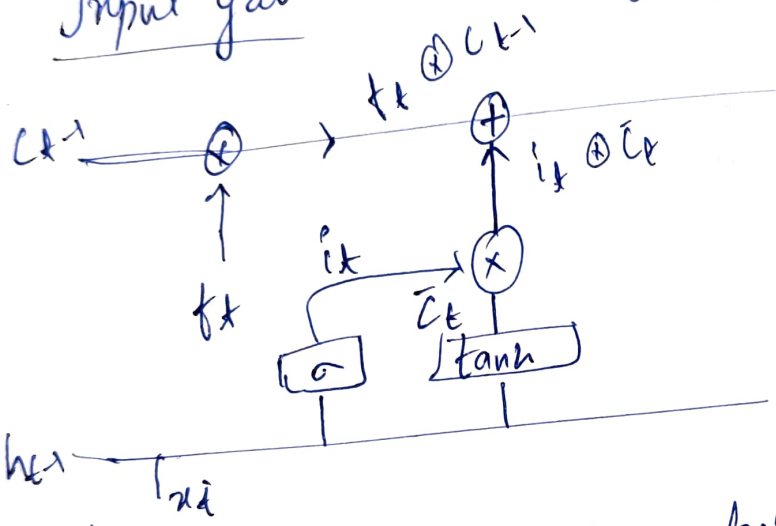
$$f_t = \sigma\left[W_f \left[h_{t-1}, x_t \right] + b_f \right]$$

W_f: 1×4 ; $[h_{t-1}, x_t]$: 4×1 ; b_f: (3×1) → (3×1)

$[h_{t-1}, x_t]$ → Concatenation of 2 vectors

$f_t \otimes C_{t-1}$ → 3×1

Input gate — adds new info to the data



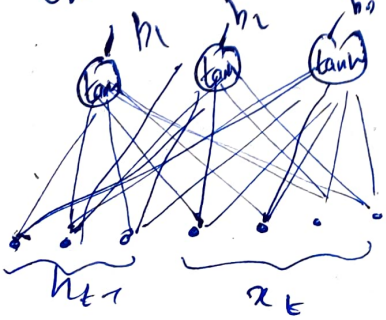
Stages:

① $\bar{C}_t$: Candidate cell state → based on current input and previous cell state, we add new info that's needed to be added.

② i_t : decide what item needed to be added to final current

③ C_t : Calculate the final cell state.

Units from NN ⇒ $\bar{C}_t$ = [ ]



→ W_R (3×4)

$\tanh\left[W_c \left(h_{t-1}, x_t \right) + b_c \right]$

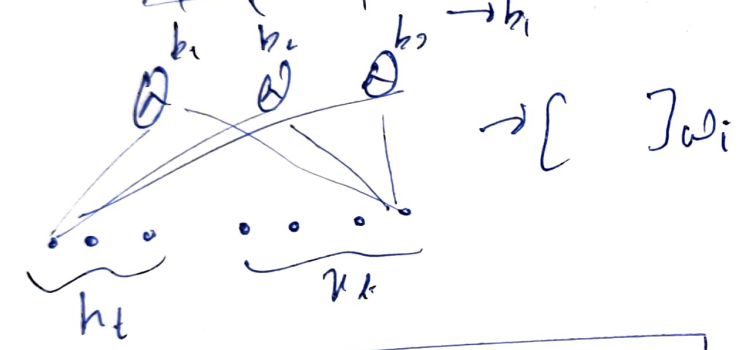
$$\bar{C}_t = \tanh\left[W_c (h_{t-1}, x_t) + b_c \right]$$

$(3+7)$ $(7+1)$ (3×1) (3×1) (3×1)

Potential important info = $\bar{C}_t$

it filter from $\bar{C}_t$, next step, calculating i_t

activation – sigmoid $\rightarrow i_t$



$$i_t = \sigma\left[W_i [h_{t-1}, x_t) + b_i \right]$$

(3×7) (7×1) (3×1) (3×1) 3×1

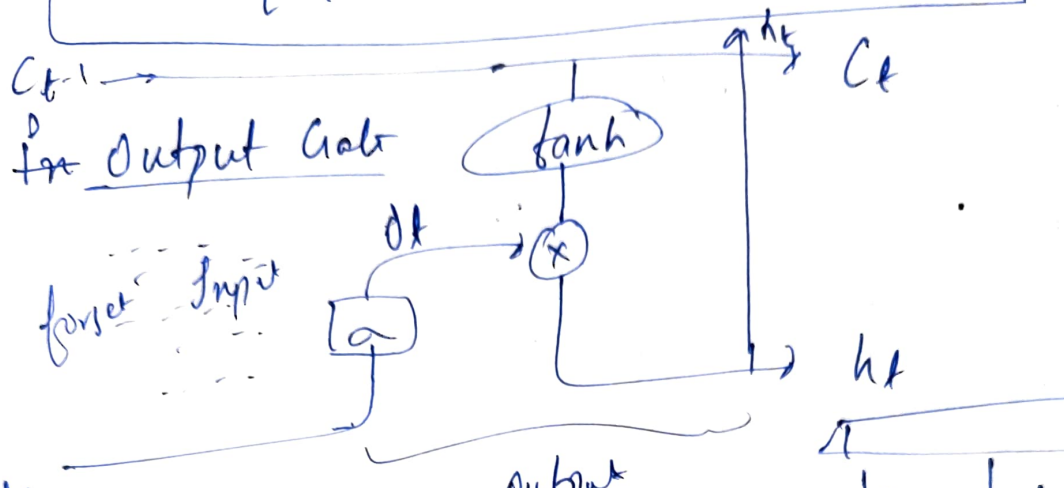
$$i_t \otimes \bar{C}_t \Rightarrow \bar{C}_t^*$$

(3×1) (3×1) → filtered Candidate Cell State

Now, finally $C_t = \left[f_t \otimes C_{t-1} \right] \oplus \left[i_t \otimes \bar{C}_t \right)$

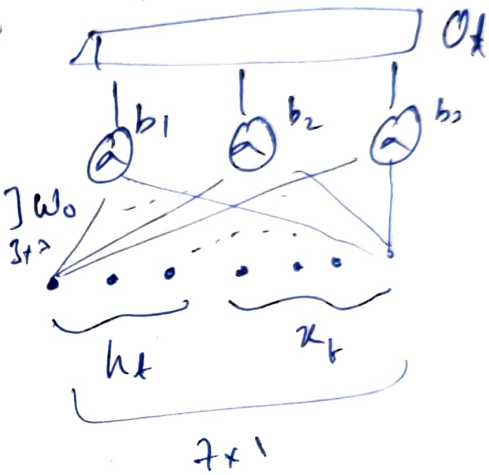
$= C_t$

$$C_t = [f_t \otimes C_{t-1}] \oplus [i_t \otimes \tilde{C}_t]$$



$$o_t \otimes \tanh(C_t)$$

$$o_t = a[w_0 [h_{t-1}, x_t] + b_0]$$



3×7 · 7×1 → 3×1

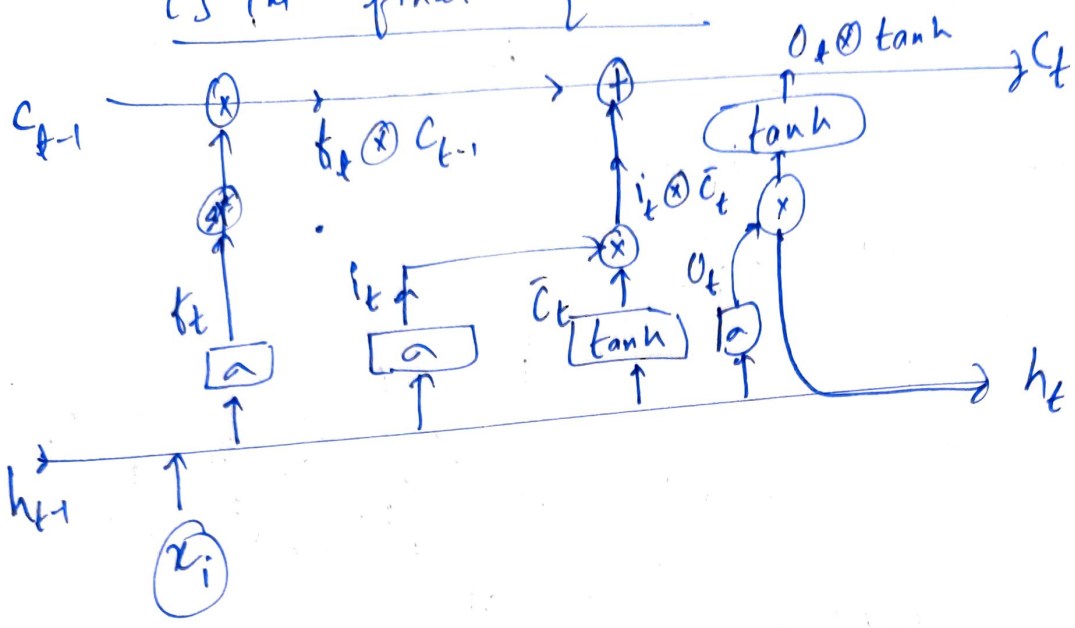
3×1 → (3×1)

a (3×1) = (3×1)

$$h_t = o_t \otimes \tanh(C_t) \rightarrow 3\times1$$

3×1 · 3×1 → 3×1

LSTM final equations



① $f_t = \sigma\left[w_f\left(h_{t-1}, x_t\right) + b_f\right)$

② $\bar{C}_t = \tanh\left[w_c\left(h_{t-1}, x_t\right) + b_c\right]$

③ $i_t = \sigma\left[w_i\left[h_{t-1}, x_t\right) + b_i\right]$

④ $C_t = \left[f_t \otimes C_{t-1}\right] \oplus \left[i_t \otimes \bar{C}_t\right]$ #

⑤ $O_t = \sigma\left[w_o\left[h_{t-1}, x_t\right] + b_o\right)$

⑥ $h_t = O_t \otimes \tanh\left[C_t\right]$ #

here:-

$[h_{t-1}, x_t]$ → concatinated Array

$\otimes$, $\oplus$, tanh → Pointwise multiplication

Attention [is all you need!]

Used to remember longer sentences

$$\text{cos similarity} = \frac{\sum A_i B_i}{\sqrt{\sum A_i^2}\ \sqrt{\sum B_i^2}}$$

Self attention :-

From each input vectors [word embeddings]

- Create 3 vectors
- 1. Query vector → $x \cdot W_Q$
- 2. Key vector → $x \cdot W_K$
- 3. Value vector → $x \cdot W_V$

} self attention using dot Product-

then → self attention score = $Q_1 \cdot K_1$ (Key vector / Query vector)

↓

do this for each word

$$Z = \left[\text{Softmax}\left[\frac{Q_1 \cdot k_1}{8\sqrt{d_k}} \right] \right] \times V_1 = Z$$

layer Norm : $X + Z$

Positional Encoding

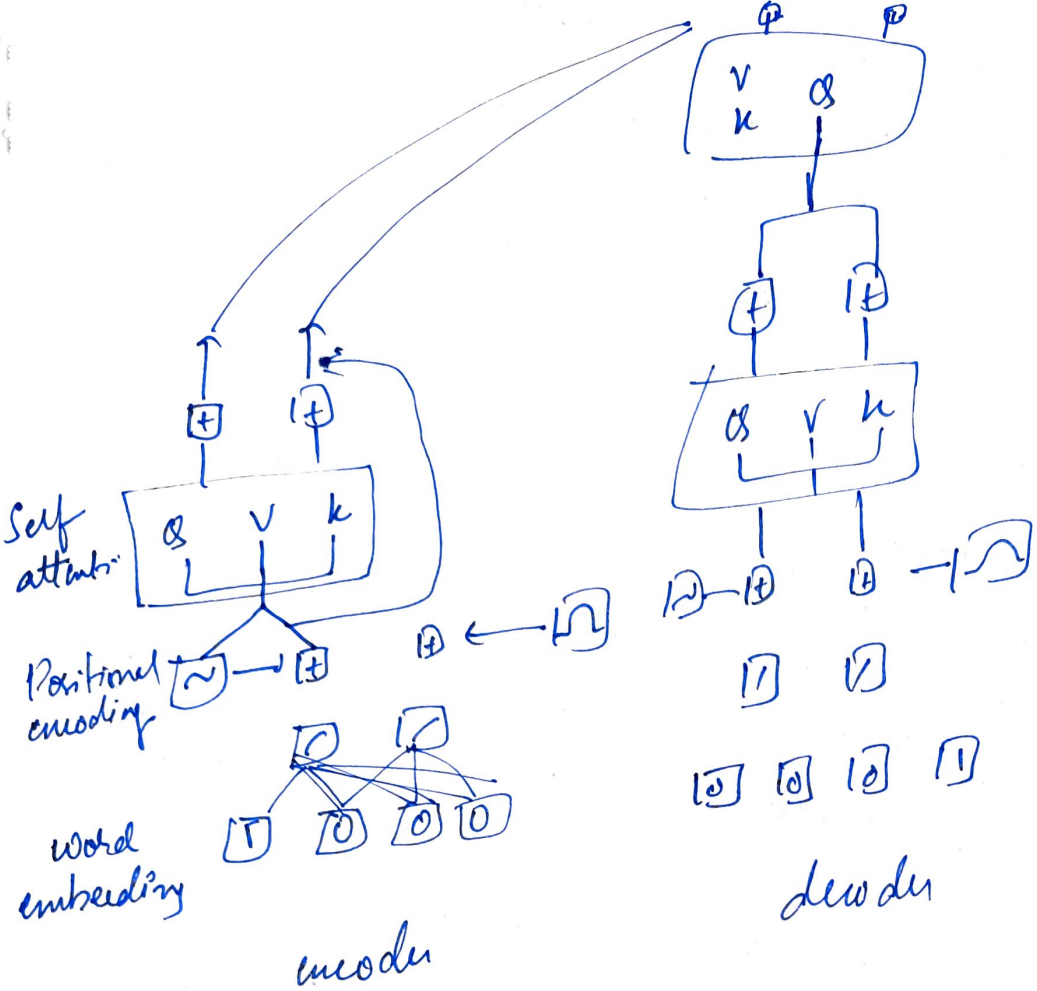
↓

self attention

↓

Add and Normalize

↓ feed forward Network → add and normal.



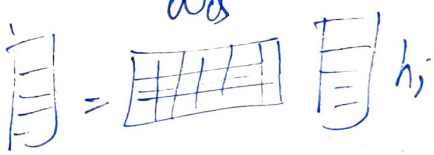
GS 6910 / CS 7015 Attention is all you need

they introduced new attention function.

$$score(S_{t-1}, h_j) = v_a^T \tanh\left[U_{att} S_{t-1} + W_{att} h_j\right]$$

Key Vector · Value Vector · Query Vector

$k_j = W_k h_j$ $\quad v_j = W_v \cdot h_j$ $\quad q_j = W_Q \cdot h_j$



$$x = \frac{x'}{\sqrt{m}} \qquad \hat{x}_{ij} = \frac{1}{\sqrt{m}} \left[x_{ij} - \frac{1}{m} \sum_{k=1}^{m} x_{kj} \right]$$

Covariance matrix $= x^T x = \frac{1}{m} (x')^T x'$

$$\frac{1}{m} \sum \sum (x_{ij} - \hat{x}_{ij})^2$$

decode = linear
loss = sq. err
encode = linear

$$\min \sum \sum (x_{ij} - \hat{x}_{ij})^2$$

$$\min \left(\|x - HW^*\|_F \right)^2$$

$$H = U_{:,:k} \Sigma_{k,k}$$

$$= (x x^T)(x x^T)^{-1} U_{:k} \Sigma_{kk}$$

$$= (x V \Sigma^T U^T)(U \Sigma V^T V \Sigma^T U^T)^{-1} U \Sigma \ldots$$

$$= x V \Sigma^T U^T U (\Sigma \Sigma^T)^{-1} U^T U \Sigma \ldots$$

$$= x V \Sigma^T (\Sigma \Sigma^T)^{-1} U^T U \ldots \Sigma \ldots$$

$$= x V \Sigma^T \Sigma^{-T} \Sigma^{-1} U^T U$$

$$= x V \Sigma^{-1} I_{\ldots k} \Sigma_{k,k}$$

$$= x V_{1 \ldots k}$$

$$H = X V_{\ldots k}$$

Convolutional Neural Networks

ANN – matrix multiplication

CNN – convolutional operation

has grid-like topology (1D data / 2D data)
↓ time series ↘ Image

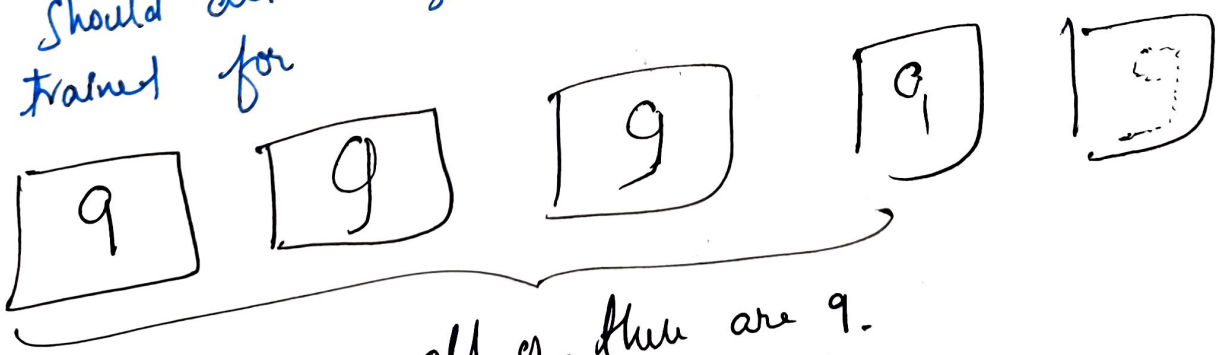
Inspired by human visual cortex

History – found by Yann Lee Cunn (bank cheques)
then microsoft did OCR.

Why not ANN ? High computation cost } Image
Overfitting
loss of data (spatial arrangement of Pixel)
[distance between images will be changed]

CNN Intution

Should detect any kind of Image that the model is trained for



all of them are 9.

Convolution layer → extract features by filters

Applications

→ Object localization → where is the particular obj in a image

→ Face recognition
→ object detection
→ Image segmentation → divide a image to segment
→ Improve resolution
→ Convert b/w to color image
→ body posture.
⋮

Road map :-

→ Biological connection
→ Convolution operation
  Padding + Stride
  - Pooling layer
  - CNN vs ANN

Biological Connection

Human visual cortex

Visual cortex have 2 cells —

Simple cell (Edge detect)
↑
detect particular formate edge orientation cell or feature detector

Complex cell
↑
bigger receptive field.

Convolution layer

Conv operation

Start layer: detect edges
next layers 2. combine edges and detect complex feature
⋮ 3. completely identify image

types
Vertical edge detection } done by kernels / filters
Horizontal " "
Blur " "
⋮
Sharp " "

edges are intensity changes.

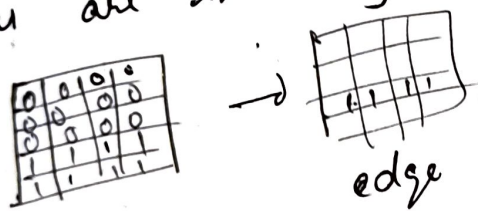


Image
Filter
Feature Map

Elem wise multiplication with kernel

Output dimension after applying kernel:

Image	kernel	output
~~n×n~~ n	m×m	(n−m+1) × (n−m+1)
n×n×c	n×n×c	(n−n+1) × (n−n+1) Single channel

Padding and Strides

When every layer the size of image keeps reducing which is not a optimal process so we use padding and strides and boundary pixels will be very less part of the convolution operation rather than the middle pixel

after padding, image size = $(n + 2p - f + 1)$

5 x 5 image

3 x 3 kernel

Padded = $5 + 2(1) - 3 + 1$

coding part

Padding = "Valid" → this means padding is not used

"same" → will keep padding (0)

Stride

when applying filter on the image, it moves the filter one pixel if stride is kept 1.

If stride is (2 x 2) then the filter will move 2 pixels

Stride 1, then dimension of image is decreased $(n - f + 1)$

[stride 1, dimension ↓] $\left[\frac{n + 2p - f}{2} + 1\right]$

Strided convolution

- Stride value other than 1

n = image

f = filter

p = padding

Stride = 1 → default stride

Stride ≠ 1 → Strided convolution

s = stride

$\left[\frac{n - f}{s} + 1\right]$ → floor operation

Why stride | Strided convolution ?

→ So that the model don't miss the info and
→ need high level feature, no need of low level features
then can use stride > 1
→ for computing power.

$$\left[\frac{n + 2p - f}{s} + 1\right]$$

Pooling

convolution has issues

1. Memory
2. Translation Variance & location independency

Way to down sample feature map
Applied after Non linear feature map is given
Type - max, min -- etc
Advantages :- ① feature map size reduces
② Translation Invariance.
Disadvantage :- ① Can't use in Image Segmentation
② lose lot of information (75%)
(depends on application.

Architecture

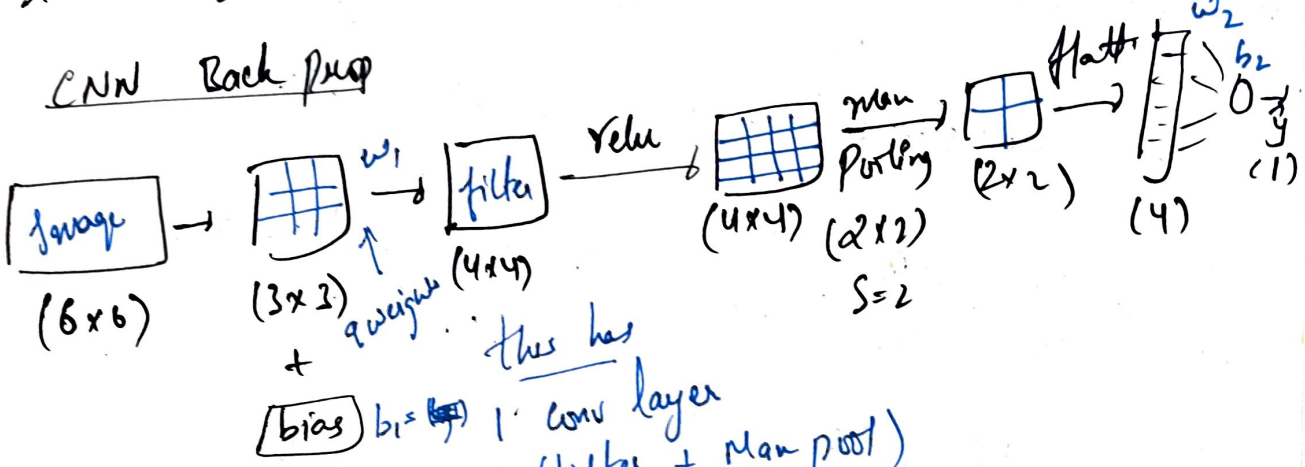
convolution → padding/stride → Pooling

CNN vs ANN

both are kinda same but small multiplication

fine differences

CNN Back prop



this has
1 conv layer
1 (filter + max pool)

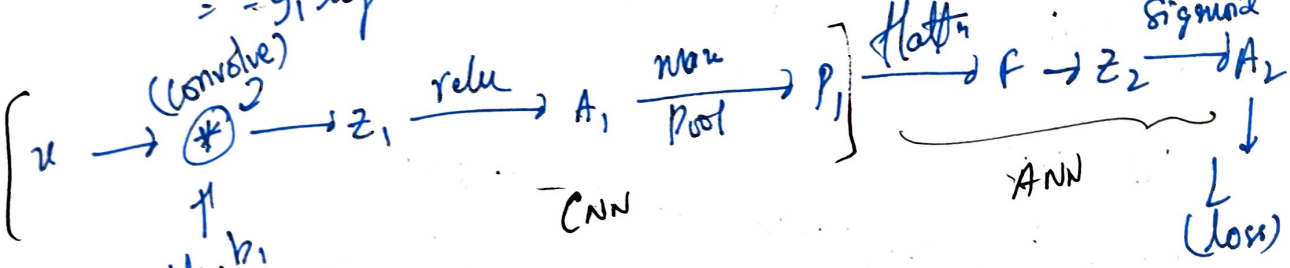
Trainable parameters

$w_1 = (3,3)$ $w_2 = (1,4)$ } total = 15 parameters

$b_1 = (1,1)$ $b_2 = (1,1)$

loss = binary classification (like cat/dog)

$= -y_1 \log(\hat{y}_1) - (1-y_1) \log(1-\hat{y}_1)$



Forward propagation :

$z_1 = \text{conv}(x, w_1) + b_1$

$A_1 = \text{relu}(z_1)$

$P_1 = \text{maxpool}(A_1)$

$F = \text{flatten}(P_1)$

$z_2 = w_2 \cdot F + b_2$

$A_2 = \sigma(z_2)$

Shapes

$F = (4,1)$

$w_2 = (1,4)$

$z_2 = (1,1)$

$A_2 = (1,1)$

$w_2 \cdot F$

$(1,4) \quad (4,1) = (1,1)$

end goal : Apply Gradient Descent on the final output and find out loss and minimize it

Gradient descent :-

matrix

$$w_1 = w_1 - \eta \frac{\partial L}{\partial w_1} \quad ; \quad w_2 = w_2 - \eta \frac{\partial L}{\partial w_2}$$

matrix

$$b_1 = b_1 - \eta \frac{\partial L}{\partial b_1} \quad ; \quad b_2 = b_2 - \eta \frac{\partial L}{\partial b_2}$$

divide backprop into 2 parts ① CNN ② ANN

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial A_2} \cdot \frac{\partial A_2}{\partial z_2} \cdot \frac{\partial z_2}{\partial w_2} \quad [\because w_2 \text{ depend on } A_2, z_2] \quad \text{—— ①}$$

$$\frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial A_2} \cdot \frac{\partial A_2}{\partial z_2} \cdot \frac{\partial z_2}{\partial b_2} \quad \text{—— ②}$$

first lets see w_1, b_1 → flatten → max pooling

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial A_2} \cdot \frac{\partial A_2}{\partial z_2} \cdot \frac{\partial z_2}{\partial F} \times \frac{\partial F}{\partial P_1} \cdot \frac{\partial P_1}{\partial A_1} \cdot \frac{\partial A_1}{\partial z_1} \cdot \frac{\partial z_1}{\partial w_1} \quad \text{—— ③}$$

$$\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial A_2} \cdot \frac{\partial A_2}{\partial z_2} \cdot \frac{\partial z_2}{\partial F} \cdot \frac{\partial F}{\partial P_1} \cdot \frac{\partial P_1}{\partial A_1} \cdot \frac{\partial A_1}{\partial z_1} \cdot \frac{\partial z_1}{\partial b_1} \quad \text{—— ④}$$

---

lets solve now :-

$$\frac{\partial L}{\partial a_2} = \frac{\partial}{\partial a_2} [-y_i \log(a_2) - (1-y_i) \log(1-a_2)]$$

$$= \frac{-y_1}{a_2} + \frac{1-y_1}{1-a_2} = \frac{-y_1(1-a_2) + a_2(1-y_1)}{a_2(1-a_2)}$$

$$\boxed{\frac{\partial L}{\partial a_2} = \frac{a_2 - y_1}{a_2(1-a_2)}}$$

$$\frac{\partial A_2}{\partial Z_2} = a(Z_2)\left[1 - a(Z_2)\right] = a_2\left[1 - a_2\right]$$

$$\boxed{\frac{\partial A_2}{\partial Z_2} = a_2\left(1 - a_2\right)}$$

$$\boxed{\frac{\partial Z_2}{\partial W_2} = F}$$

$$\boxed{\frac{\partial Z_2}{\partial b_2} = 1}$$

$$\frac{\partial L}{\partial W_2} = \frac{\partial A_2}{\partial Z_2} \cdot \frac{\partial Z_2}{\partial W_2} \cdot \frac{\partial Z_2}{\partial W_2}$$

$$\frac{\partial L}{\partial W_2} = \frac{a_2 - y_1}{a_2(1-a_2)} \times \underset{(1,1)\ (1,1)}{a_2(1-a_2)} \times \underset{(1,4)\ (4,4)}{F} =$$

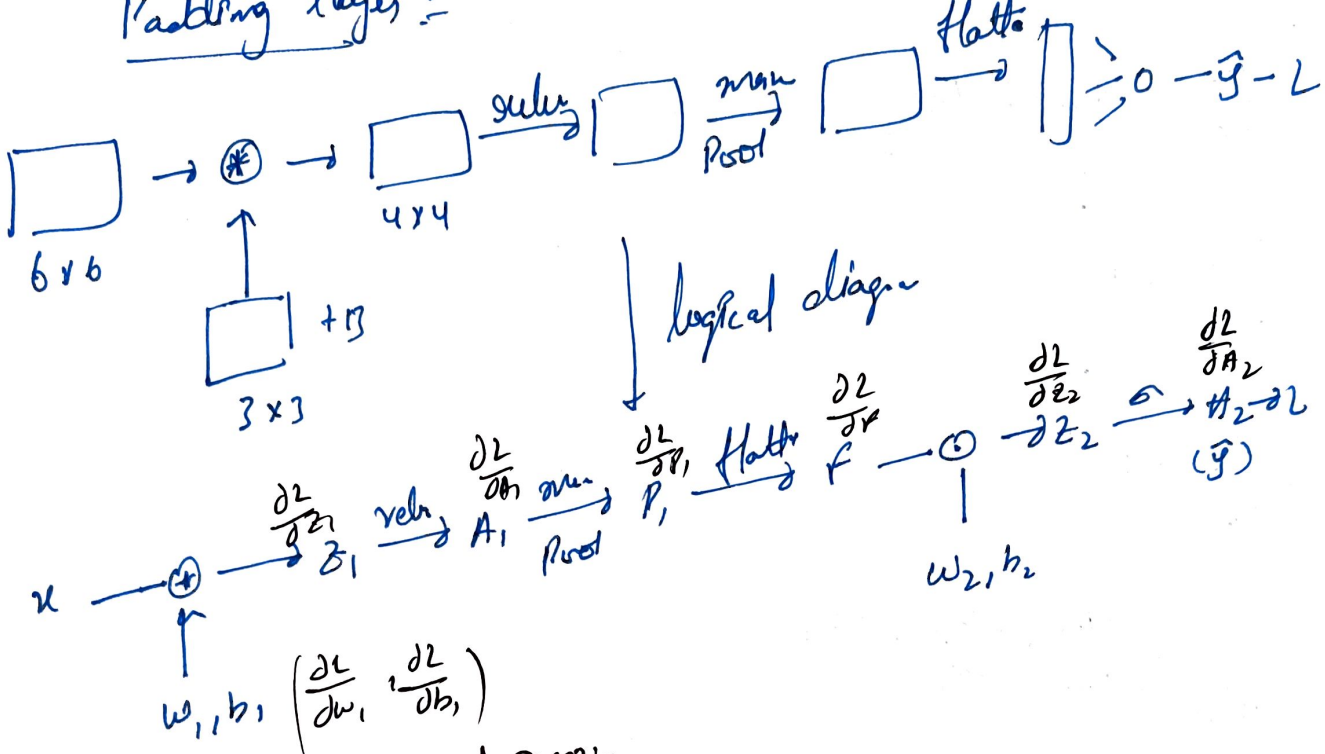
$$= (a_2 - y_1) F = (A_2 - y) F$$

$$\boxed{\frac{\partial L}{\partial W_2} = (A_2 - y) F^T} \quad \text{Shape} = (1,4)$$

$$\frac{\partial L}{\partial b_2} = \frac{a_2 - y_1}{a_2(1-a_2)} \quad a_2(1-a_2) \times 1$$

$$\boxed{\frac{\partial L}{\partial b_2} = (A_2 - y)}$$

Now backprop for convolution layer, max pool layer and Padding layer :-



Equations for forward prop :-

$Z_1 = \text{conv}(x, w_1) + b_1$

$A_1 = \text{relu}(Z_1)$

$P_1 = \text{max pool}(A_1)$

$F = \text{Flatten}(P_1)$

$Z_2 = w_2 F + b_2$

$A_2 = \sigma(Z_2)$

$L = \frac{1}{m} \sum_{i=1}^{m} [-y_i \log(A_2) - (1-y_i) \log(1-a_2)$

Should Calculate ③ and ④

$\frac{dZ_2}{dF} = \frac{d}{dF}(w_2 F + b_2) = w_2$

$\boxed{\frac{dZ_2}{dF} = w_2}$ ↳ Should reshape all the terms ↳ Shape =? ! Same as F

$\frac{\partial F}{\partial P_1} = \frac{d}{dP_1}[\text{flatten}(P_1)]$ → Should reshape this to its original matrix size

$$\frac{\partial F}{\partial P_1} = \text{reshape}(P_1, \text{Shape})$$

$$\frac{\partial L}{\partial w_1} = (A_2 - y)\, w_2 \cdot \text{reshape}(P_1 \cdot \text{Shape}) \frac{\partial P_1}{\partial A_1} \cdot \frac{\partial A_1}{\partial Z_1} \cdot \frac{\partial Z_1}{\partial w_1}$$

So $\frac{\partial P_1}{\partial A_1} = \frac{\partial}{\partial A_1} \text{maxpool}(A_1)$ → Should do reverse of maxpooling

Only max elements will have a role in loss Prediction

$$\begin{bmatrix} x_1 & x_2 \\ x_3 & x_4 \end{bmatrix}_{2\times 2} \rightarrow \begin{bmatrix} 0 & x_1 & x_2 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & x_3 & x_4 & 0 \end{bmatrix}_{4\times 4}$$

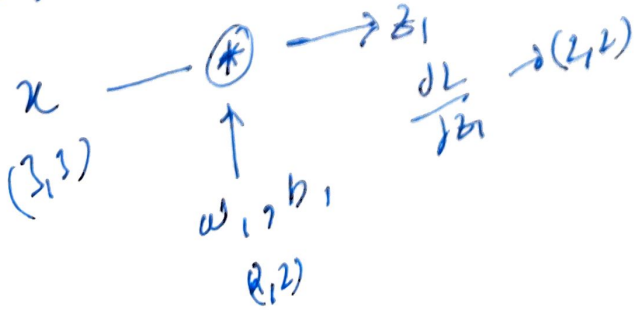
(4×4) matr. ~~(2×2)~~ (2×2)

replacing x_1, x_2, x_3, x_4 to their original place ~~at~~ before max pooling is done

$$\frac{\partial L}{\partial A_1} = \begin{cases} \frac{\partial L}{\partial P_1 xy} & \text{if } A_{mn} \text{ is the max element} \\ 0 & \text{if, otherwise} \end{cases}$$

$$\frac{\partial A_1}{\partial Z_1} = \begin{cases} 1 \text{ ; if } Z_1 xy > 0 \\ 0 \text{ ; if } Z_1 xy < 0 \end{cases}$$

$\frac{\partial Z_1}{\partial w_1}$ → convolution backprop, thus has trainy parameter



$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} \quad W_1 = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} \quad Z_1 = \begin{bmatrix} z_{11} & z_{12} \\ z_{21} & z_{22} \end{bmatrix}$$

$$z_{11} = x_{11} w_{11} + x_{12} w_{12} + x_{21} w_{21} + x_{22} w_{22} + b_1$$

$$z_{12} = x_{12} w_{11} + x_{13} w_{12} + x_{22} w_{21} + x_{23} w_{22} + b_1$$

$$\frac{\partial L}{\partial Z_1} = \begin{bmatrix} \frac{\partial L}{\partial z_{11}} & \frac{\partial L}{\partial z_{12}} \\ \frac{\partial L}{\partial z_{21}} & \frac{\partial L}{\partial z_{22}} \end{bmatrix}$$

$$z_{21} = x_{21} w_{11} + x_{22} w_{12} + x_{31} w_{21} + x_{32} w_{22} + b_1$$

$$z_{22} = x_{22} w_{11} + x_{23} w_{12} + x_{32} w_{21} + x_{33} w_{22} + b_1$$

$$\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial z_1} \cdot \frac{\partial z_1}{\partial b_1} = \frac{\partial L}{\partial z_{11}} \cdot \frac{\partial z_{11}}{\partial b_1} + \frac{\partial L}{\partial z_{12}} \cdot \frac{\partial z_{12}}{\partial b_1} + \frac{\partial L}{\partial z_{21}} \cdot \frac{\partial z_{21}}{\partial b_1}$$

$$+ \frac{\partial L}{\partial z_{22}} \cdot \frac{\partial z_{22}}{\partial b_1}$$

$$= \frac{\partial L}{\partial z_{11}} + \frac{\partial L}{\partial z_{12}} + \frac{\partial L}{\partial z_{21}} + \frac{\partial L}{\partial z_{22}} = \text{Sum}\left(\frac{\partial L}{\partial z_1}\right)$$

$$\boxed{\frac{\partial L}{\partial b_1} = \text{Sum}\left(\frac{\partial L}{\partial z_1}\right)} \quad \text{Scalar}$$

V.A.E

11.28.23

Practice (for own)

Variational Auto Encoders

normal encoder { Used for compressing
Used for segmentation in self driving Cars
Reconstructing parts of a image

VAE is encoded to distribution
mean vector — Standard deviation

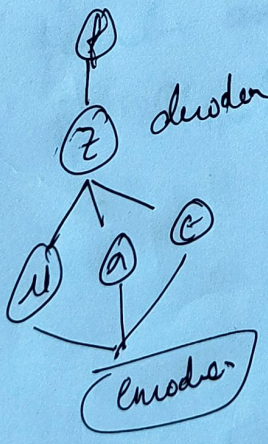
loss = Reconstruction loss + KL divergence

Sampling : latent Vector.
Cant push gradients to this Sample
hence backprop is hard.
Reparameterization trick.

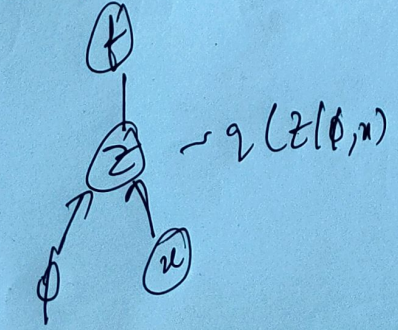
$z = \mu + \sigma \odot \epsilon$

$\epsilon \sim Normal(0,1)$
0 mean
1 S.D

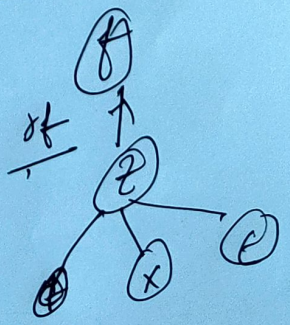
(0,1) btw close to (0,1)



Original

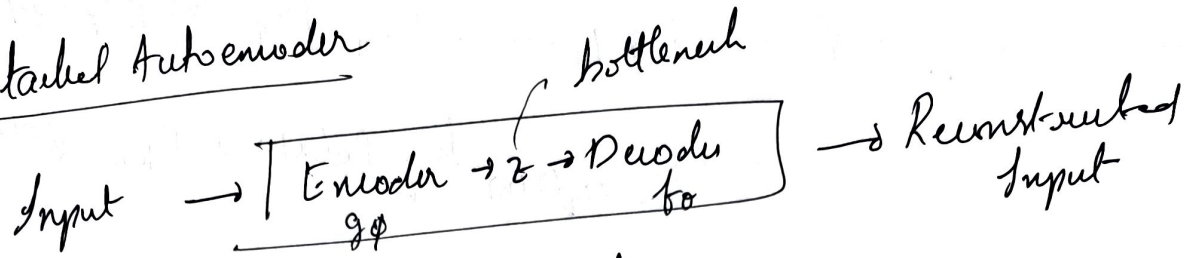


Reparameterizational form



VAE by Aladad Kumar on youtube.

Stacked Autoencoder



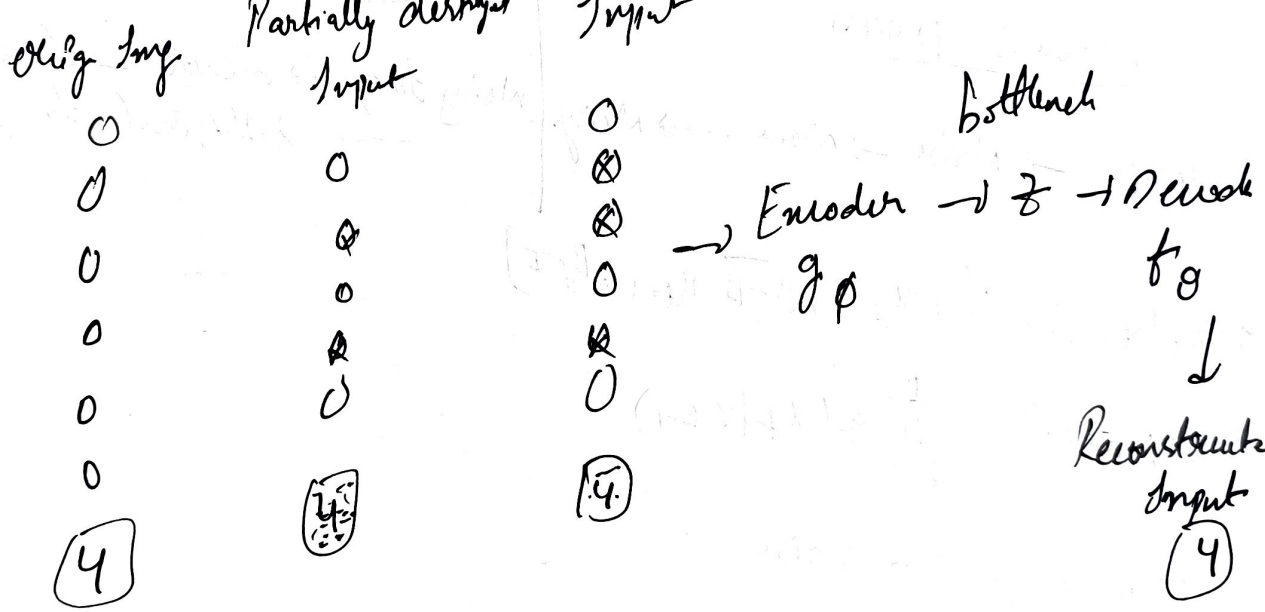
θ, ϕ : parameters

cost function = $L(\theta, \phi)$

$$L(\theta,\phi) = \frac{1}{n}\sum_{i=1}^{n}\left[x^i - f_\theta\left[g_\phi(x^i)\right]\right]^2$$

Reconstructed Image

Denoising Autoencoder :-

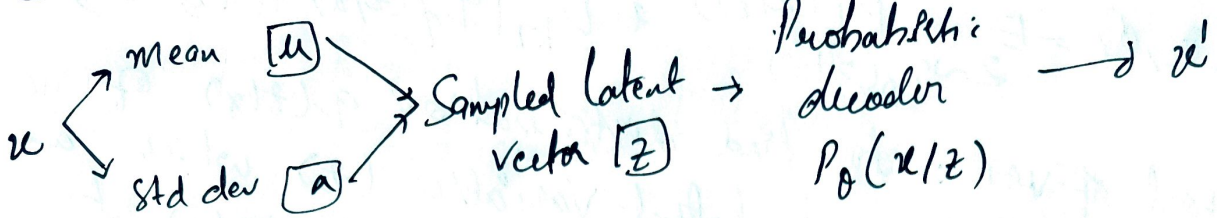


$$L(\theta,\phi) = \frac{1}{n}\sum_{i=1}^{n}\left[x^i - f_\theta\left[g_\phi(\tilde{x}^i)\right]\right]^2$$

$$\tilde{x}^i \sim x\left[\tilde{x}^i \mid x^i\right]$$

VAE's

$q_\phi(z|x)$



Probabilistic Encoder

$z = \mu + \sigma \odot \epsilon$

$\epsilon \sim \mathcal{N}(0, I)$

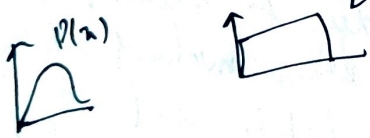
$$L(\theta, \phi) = -E_{z \sim q_\phi(z|x)}\left[p_\theta(x|z)\right] + D_{KL}\left[q_\phi(z|x) \,||\, p_\theta(z)\right]$$

$E(x) = \sum_{i=1}^{n} x\, P(x)$ ——— expectation.

KL Divergence (Kullback Leibler)

used to measure how 1 p.d is different from other p.d.

p.d = probabilistic distribution



KLD is high for this.

KLD is almost or close to 0.

$$D_{KL}(P||Q) = \sum P(X=x) \log\left[\frac{P(X=x)}{Q(X=x)}\right]$$

$$= \sum P(x) \log\left[\frac{P(x)}{Q(x)}\right]$$

what are VAE's and why we need it?

$$L(\theta, \phi) = E_{z \sim q_\phi(z|x)} P[(x|z)] + D_{KL}[q_\phi(z|x) \,||\, P_\theta(z)]$$

goal of VAE: to find distribution $q(z|x)$ of some latent variables (z) which we can sample from $z \sim q_\phi(z|x)$ to generate new samples $x' \sim P_\theta(x|z)$

lower

Approximate Inference

x - observed variable

z - latent variable

$$p(z|x) = \frac{p(x|z) \cdot p(z)}{p(x)} \quad \text{(conditional prob)} \quad (A)$$

difficult to obtain $p(x)$

$p(x) = \int p(x|z)\, p(z)\, dz$ → not closed integral or intractable due to multiple integral are involved

alternative?

approximate $p(z|x)$ by $q(z|x)$ ~ has tractable sol

done by variational Inference

Posing Inference problem as Optimization problem by modelling $p(z|x)$ using $\Phi(z|x)$ which has simple Gaussian distribution

$$D_{KL}\left[Q_\phi(z|x) \,\|\, P_\theta(z|x)\right] = \sum Q(z|x) \log\left[\frac{Q(z|x)}{P(z|x)}\right]$$

Q_ϕ: gaussian; P_θ: one we are trying to approximate

$$= E_{z\sim Q(z|x)}\left[\log\left[\frac{Q(z|x)}{P(z|x)}\right]\right]$$

$$D_{KL}\left[Q_\phi(z|x) \,\|\, P_\theta(z|x)\right] = E_{z\sim Q(z|x)}\left[\log\left(Q_\phi(z|x)\right) - \log\left(P_\theta(z|x)\right)\right] \quad (R)$$

Substituting (A) in (B), we get

$$D_{KL}\left[Q_\phi(z|x) \| P_\theta(z|x)\right] = E_{(z)}\left[\log Q_\phi(z|x) - \log \frac{P(x|z)\,P(z)}{P(x)}\right]$$

$$= E_{(z)}\left[\log Q_\phi(z|x) - \log P_\theta(x|z) - \log P_\theta(z) + \log P_\theta(x)\right]$$

z is being sampled by $z\sim Q_\phi(z|x)$ = (z)

Since expectation is over z, $P(x)$ doesn't involve z, it can be moved out.

$$D_{KL}\left[Q_\phi(z|x) \| P_\theta(z|x)\right] - \log P_\theta(x) = E_{(z)}\left[\log Q_\phi(z|x) - \log P_\theta(x|z) - \log P_\theta(z)\right]$$

Rearranging equations :-

$$\log P_\theta(x) - D_{KL}\left[Q_\phi(z|x) \| P_\theta(z|x)\right] = E_{(z)}\left[\log P(x|z)\right] - E_{(z)}\left[\log Q(z|x) - \log P(z)\right]$$

this is basically KL divergence between $Q(z|x)$ and $P(z)$

$$= E_{(z)}\left[\log P(x|z)\right] - D_{KL}\left[Q(z|x) \| P(z)\right]$$

this is VAE objective function :-

1st term = Reconstruction likelihood

2nd term = ensures learned distribution Q is similar to prior distribution P

loss = - [objective function]

$$L = L(\theta, \phi) = -E_{z \sim q_\phi(z|x)}[\log p_\theta(x|z)] + D_{KL}[q_\phi(z|x) \| p(z)]$$

Also :- hence, proved.

$$\log p_\theta(x) - D_{KL}[q_\phi(z|x) \| p(z|x)] = -L(\theta, \phi)$$

Reconstruction loss. — Inherent Regularizer

$$\theta^*, \phi^* = \arg\min L(\theta, \phi)$$

→ optimal θ, ϕ such that

So our target is to find

weights of encoder. — weights of generative model (decoder)

Reconstruction loss

$$-E_{z \sim q_\phi(z|x)}[\log p_\theta(x|z)]$$

Regularizer

$$D_{KL}[q_\phi(z|x) \| p_\theta(z)]$$

What are these both?

↳ log likelihood [alias] Recognition Model

$$p_\theta(x|z) = N(\psi_\theta(z), \Sigma_\theta(z))$$

for this we get squared error between data sample and mean of gaussian distribution.

$$p_\theta(x|z) = \frac{1}{\sqrt{2\pi^k |\Sigma_\theta(z)|}} \exp\left[\frac{[x - p_\theta(z)]^T \Sigma_\theta^{-1}(z) (x - p_\theta(z))}{2}\right]$$

$$\log p_\theta(x|z) \propto \underbrace{x - p_\theta(z)^T \Sigma_\theta^{-1}(z) (x - p_\theta(z))}_{\text{Squared Reconstruction Error [data fedility term] in inverse Problem}}$$

VAE derivation

$$P_\theta(z|x) = \frac{P(x|z)\,P(z)}{P(x)} \quad \text{VAE} \quad (4) \qquad (1)$$

$$D_{KL}\left[Q(z|x) \,\|\, P(z|x)\right] = \sum Q(z|x) \cdot \log \frac{Q(z|x)}{P(z|x)}$$

$$= E_{z}\left[\log\left[\frac{Q(z|x)}{P(z|x)}\right]\right]$$

$$D_{KL}\left[Q(z|x) \,\|\, P(z|x)\right] = E_{z}\left[\log\left(Q(z|x)\right) - \log P(z|x)\right] \quad (B)$$

Putting (A) in (B)

$$= E_{z}\left[\log Q(z|x) - \log\left[\frac{P(x|z)\,P(z)}{P(x)}\right]\right]$$

$$= E_{z}\left[\log Q(z|x) - \log\left(P(x|z)\,P(z)\right) + \log P(x)\right]$$

$$= E_{z}\left[\log Q(z|x) - \log P(x|z) - \log P(z) + \log P(x)\right]$$

$Q = Q_\phi$; $P = P_\theta$; $z = z \sim Q_\phi(z|x)$

$\log P(x)$ can be moved out as it doesn't involve z

$$D_{KL}\left[Q(z|x) \,\|\, P(z|x)\right] - \log P(x) = E_{z}\left[\log Q(z|x) - \log P(x|z) - \log P(z)\right]$$

$$\log P(x) - D_{KL}\left[Q(z|x) \,\|\, P(z|x)\right] = E_{z}\left[\log P(x|z)\right] - E_{z}\left[\log Q(z|x) - \log P(z)\right]$$

$$= E_{z}\left[\log P(x|z)\right] - D_{KL}\left[Q(z|x) \,\|\, P(z)\right]$$

Reconstruction likelihood

Ensures Q (learned distribution) similar to P (prior distribution)

VAE objective fn

loss = - objective fn

$$L(\theta,\phi) = -E[\log P(x|z)] + D_{KL}[\log Q(z|x) \,||\, \log P(z)]$$

or

$$-L(\theta,\phi) = \log P_\theta(x) - D_{KL}[\log Q(z|x) \,||\, P(z|x)]$$

$D_{KL}[Q_\phi(z|x) \,||\, P_\theta(z)]$ Regularizer

$P_\theta(z)$ = latent variable distribution

Easiest choice = $N(0,1)$

Want this to be close to $Q(z|x)$ to sample it easily ($P_\theta(z)$)

If we want $Q_\phi(z|x)$ to be gaussian with parameters $\mu_\phi(x)$, $\Sigma_\phi(x)$ then the KL divergence has the closed form as derived when $P_\theta(z) = N(0,1)$

$$D_{KL}[N(\mu(x),\Sigma(x)) \,||\, N(0,1)] = \frac{1}{2}\left[\text{tr}(\Sigma(x)) + \mu(x)^T(\mu(x) - k - \log|\Sigma(x)|)\right] \quad (2)$$

k = dimensionality of latent variables

$\text{tr}(\Sigma(x))$ = trace fn = sum of diagonal matrix of $\Sigma(x)$

Proof =

$$E_p[(x-\mu_1)^T \Sigma_2^{-1}[\mu_1-\mu_2]]$$

$$\Rightarrow [E_p[x-\mu_1]^T \Sigma_2^{-1}(\mu_1-\mu_2)]$$

$$\Rightarrow (\mu_1-\mu_1)\,\Sigma_2^{-1}(\mu_1-\mu_2)$$

$$\Rightarrow 0 \quad \text{(Proved)}$$

$$KL[P(x) \,||\, Q(x)] = \frac{1}{2}\left[\log\left[\frac{\Sigma_2}{\Sigma_1}\right] - k + \text{tr}(\Sigma_2^{-1}\Sigma_1) + (\mu_1-\mu_2)^T\Sigma_2^{-1}(\mu_1-\mu_2)\right] \quad (1)$$

$P(x)$: μ_1, Σ_1 ; $Q(x)$: μ_2, Σ_2

$\Sigma_2 = I$ | $\Sigma_1 = \Sigma_\phi(x)$

$\mu_2 = 0$ | $\mu_1 = \mu_\phi(x)$

by putting these values in (1) we get (2) (main fn)

$$D_{KL}\left[N(\mu_\phi(x), \Sigma(x)) \,\|\, N(0,1)\right] =$$ (5)

$$= \frac{1}{2}\left[\sum_k \Sigma_\phi(x) + \sum_k \mu_\phi^2(x) - \sum_k 1 - \log \prod_k \Sigma(x)\right]$$

$$= \frac{1}{2}\left[\sum_k \sum_\phi(x) + \sum_k \mu_\phi^2(x) - \sum_k 1 - \sum \log\left(\Sigma_\phi(x)\right)\right]$$

$$= \frac{1}{2}\sum_k\left[\Sigma_\phi(x) + \mu_\phi^2(x) - 1 - \log\left[\Sigma_\phi(x)\right]\right]$$

Simplified closed loop expression for KL div

Reparameterization

Optimizing loss fn:-

$$\theta^*, \phi^* = \arg\min_{\theta,\phi}\left[L(\theta,\phi)\right]$$

ELBO :- evidence lower bound

$\boxed{KL \geq 0}$ Property

$$\text{Therefore; } L(\theta,\phi) = \text{lower bound of } \log P_\theta(x)$$

$$\log P_\theta(x) - D_{KL}\left[Q_\theta(z|x) \,\|\, P(z|x)\right] = -L(\theta,\phi)$$

always ≥ 0

$$\therefore \boxed{L(\theta,\phi) \leq \log P_\theta(x)}$$

$\downarrow$ loss $\uparrow$ lower bound of P(generating real data)

Recall :-

$$L(\theta,\phi) = -E_{(z)}\left[\log P_\theta(x|z)\right] + \frac{1}{2}\sum_k\left[\exp\left(\Sigma_\phi(x)\right) + \mu_\phi^2(x) - 1 - \Sigma_\phi(x)\right]$$

alternative optimization principle

$\theta^*, \phi^* = \arg\min L(\theta, \phi)$

$\theta^* = \nabla_\theta [L(\theta, \phi)]$; ϕ = constant

$\phi^* = \nabla_\phi [L(\theta, \phi)]$;

$\theta^{**} = \nabla_\theta (L(\theta, \phi))$

the cycle keeps Repeating for certain iterations

Problem will occur while derivating wrt ϕ to learn $q_\phi(z|x)$ and $P_\theta(x|z)$ at same time

$$\min_{\theta,\phi} L(\theta,\phi) = \min_{\theta,\phi} \left\{ -E_{z}(\log P(x|z)) + \frac{1}{2} \sum \left[\exp(\Sigma_\phi(x)) + \mu_\phi^2(x) - 1 - \Sigma_\theta(x) \right] \right.$$

$\hat{\theta}_i = \nabla_\theta [L(\theta, \phi)]$

$\theta_i = \frac{1}{L} \sum_{l=1}^{L} \nabla_\theta \log P_\theta(x|z^{(l)}) + 0 \; ; z^l \sim q_\theta(z|x)$

$\hat{\phi}_i = \nabla_\phi \{L(\theta, \phi)\}$

= Can't differentiate 1st term + Can do for 2nd term

∇_ϕ is harder to estimate

So here we use reparameterization technique

i.e., $\nabla_\phi E_{q_\phi(z|x)}[f(z)] \neq E_{q_\phi(z|x)}[\nabla_\phi f(x)]$

If we can somehow rewrite the Expectation such that ϕ appears inside the expectation. Then we can differentiate

i.e.,

$$E_{q_\phi(z|x)}[f(z)] = E_{p(\epsilon)}[f(g_\phi(\epsilon, x))]$$

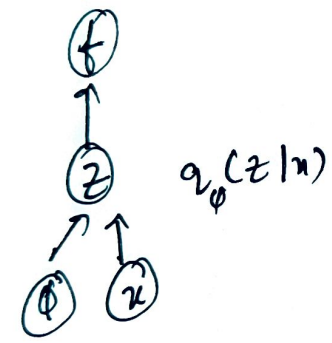
such that $z = g_\phi(\epsilon, x)) \sim \epsilon \sim N(0,1)$

linear transformation $\Rightarrow g_\phi(\epsilon, x)$

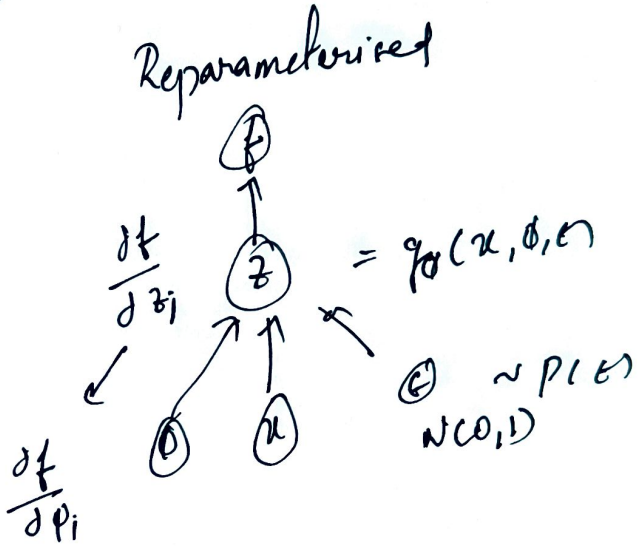
$$g_\phi(\epsilon, x) = \mu_\phi(x) + \epsilon \odot \Sigma_\phi^{1/2}(x) = z \sim N(\mu(x), \Sigma(x))$$

Mean Covariance

Original



Reparameterized



$= \partial L / d\phi_i$

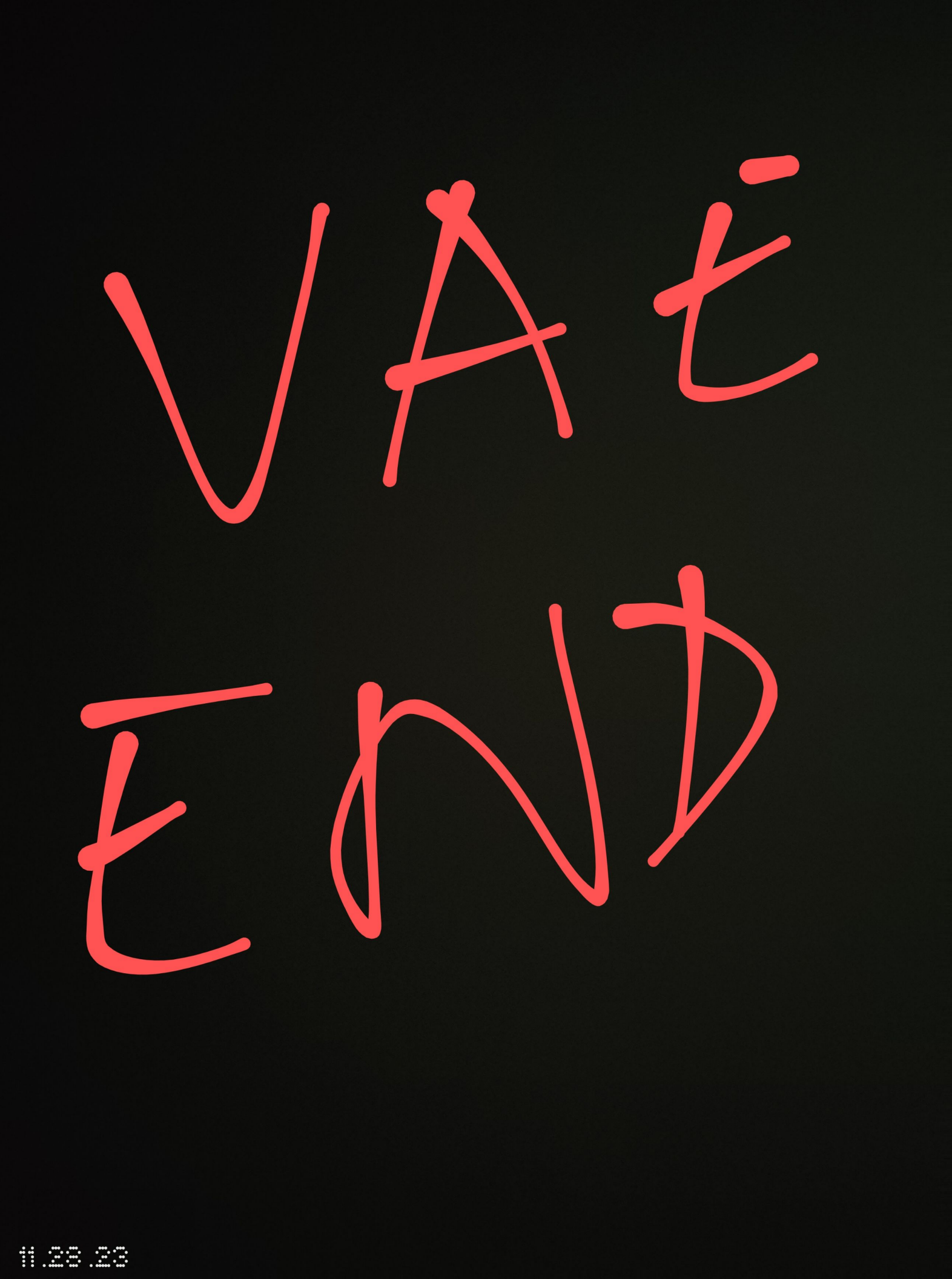
finally

$$\hat{\phi}_i = -E[\nabla_\phi(\log P_\theta(x|z)] + \nabla_\phi\left[\frac{1}{2}\left(\sum_k [\exp(\Sigma(x)) + \psi_\phi^2(x) - 1 - \Sigma(x)\right)\right]$$

$$= -\frac{1}{S}\sum_{s=1}^{S} \log P_\theta(x|z^l) +$$

Monte Carlo estimate

$z^l = \mu_\phi(x) + \epsilon \odot \sigma_\phi(x)$ where $\epsilon^l \sim N(0,1)$



11.28.23

GANs START

11.28.23

Generative model (or)

Regularizer

$D_{KL}[Q_\phi(z|x) \,||\, P_\theta(z)]$

$Q_\phi(z|x)$: Recognition model that take mapping from x and produces output z

$P_\theta(z)$: gaussian $\begin{cases} 0 \text{ mean} \\ 1 \text{ Variance} \end{cases}$ Assume as known.

We want the generated data z, not to be too far from Normal distribution ; $N(0,1)$

this is basically acting like a regularizer, that want allow the latent variables to have distribution away from $(0,1)$. (Acting like softmax activation fn)

Pdf of latent Variable → $Q_\phi(z|x)$

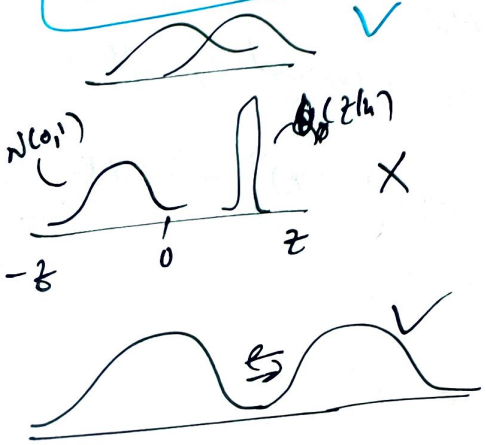
$N(0,2)$ lets it stay in this range. → $P_\theta(z)$

L = data fidelity + KL divergence

$\times$ → deviates from distribution

✓ → Network cheats by learning narrow distribution Variance is very small

✓ → Sufficient Variance is ensured attraction btw 2 is there



Simplifying Regularizer Term $[D_{KL}[Q_\phi(z|x) \| P_\theta(z)]]$

$P_\theta(z)$ = latent Variable

mean = 0

Variance = 1

→ here, we need $P_\theta(z)$ to be as close to $Q_\phi(z|x)$ to sample it easily.

→ Having $P_\theta(z) = \mathcal{N}(0,1)$ adds another benefit

optimization of loss in VAE

$\theta^*, \phi^* = \{\arg\min_{\theta,\phi} \{L(\theta,\phi)\}\}$

ELBO proof :-

$\log P_\theta(x) - D_{KL}[Q_\phi(z|x) \| P(z|x)] = -L(\theta,\phi)$

and we know ; $D_{KL}[Q_\phi(z|x) \| P_\theta(z|x)] \geq 0$ (ELBO)

hence ; $L(\theta,\phi) \leq \log P_\theta(x)$ by putting this in

In Variational Bayesian Method, this loss fn ($L(\theta,\phi)$) is known as Variational lower Bound or Evidence lower bound [ELBO]. This "lower Bound" part comes from the fact that KL Div is always Non-Negative and thus $L(\theta,\phi)$ is the lower bound of $\log P_\theta(x)$

Therefore we are minimizing the loss and maximizing lower bound of probability of generating real data samples ↳ $[L(\theta,\phi) \leq \log P_\theta(x)] \rightarrow$ ELBO

Reparameterization Trick

Problem is. doing derivatives with ϕ. Doing derivatives with respect to θ is absolutely fine.

Solution :-

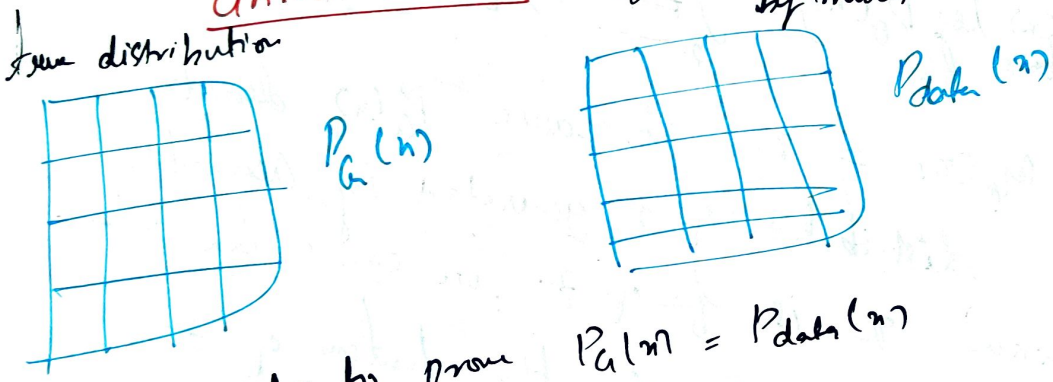
$$L(\theta,\phi) = \nabla\left[-E_{z\sim Q}[\log P_\theta(x|z)] + \frac{1}{2}\sum_k \left[\exp[\Sigma_\phi(x)] + P_\phi^2(x) - 1 - \Sigma_\phi(x)\right]\right]$$

optimization is carried out wrt θ and ϕ at same time

Run algo upto fixed number of iterations.

Mitesh Khapra

GAN's Math



we want to prove $P_G(x) = P_{data}(x)$

we dont have $P_G(x)$ formula

we can say this in different versions :-

→ if the discriminators loss is at its minimum it can happen only if $P_G(x) = P_{data}(x)$ → obj fn

If $P_G(x) = P_{data}(x)$ → global minimum is achieved

Only If $C(G) = \max_D V(G,D)$ achieved only if $P_G = P_{data}$

Step 1 :- we need to find $V(G,D)$

① find when $P_G = P_{data}$

② find when $P_G \neq P_{data}$

③ $\forall\ T :- \quad a < b \quad \forall\ P_G = P_{data}$

obj fn :-

$$\min \max \left[E_{x \sim P_{data}} \log D_\theta(x) + E_{x \sim P(z)} \log \left[1 - D_\theta(G_\phi(z)) \right] \right] \longrightarrow (1)$$

expanding to its Integral :-

$$\min \max \left[\int_x P_{data}(x) \log D_\theta(x) \right] + \int_z P(z) \log \left(1 - D_\theta(G_\phi(z)) \right)$$

this is generated x

$$\min \max \left[\int_x P_{data}(x) \log D_\theta(x) \right] + \int_z P(z) \log \left[1 - D_\theta(x) \right] \longrightarrow (2)$$

$\because G_\phi(z) = x$ because $P_G(x)$ denotes distribution of x generated by Generator

Rule → [Change of Variables. Assume it's invertable.]

Since x is fn of z, we can replace it.

here P_G is the function of z which is many to many function

Revised obj fn — ② :-

$$\min \max \int_x P_{data}(x) \log D_\theta(x) + P_G(x) \log \left[1 - D_\theta(x) \right] dx \longrightarrow (2)$$

When will it get maximum?

when every term in sum is maximized

take derivative and set it to 0

$$\frac{d}{d[D_\theta(x)]} \; P_{data}(x) \log D_\theta(x) + P_G(x) \log [1 - D_\theta(x)] = 0$$

$$P_{data}(x) \frac{1}{D_\theta(x)} + P_G(x) \frac{1}{1 - D_\theta(x)} (-1) = 0$$

$$\frac{P_{data}(x)}{D_\theta(x)} = \frac{P_G(x)}{1 - D_\theta(x)}$$

$$P_{data}(x)\,(1 - D_\theta(x)) = P_G(x)\, D_\theta(x)$$

$$D_\theta(x) = \frac{P_{data}(x)}{P_G(x) + P_{data}(x)}$$ } we get optimal discriminator

↳ it is a score given x, the best score that can be assigned.

for any given generator, the optimal discriminator is given by $D_\theta(x)$

If $P_{data} = P_G$ $\therefore$ $D_\theta(x) = \frac{P_{data}}{P_{data} + P_G} = 1/2$

half probability is real / fake

$$V(G, D) = \int P_{data}(x) \log D_\theta(x) + P_G(x) \log (1 - D(x))\, dx$$

$$= \int P_{data}(x) \log \frac{1}{2} + P_G(x) \cdot \log \left(1 - \frac{1}{2}\right) dx$$

$$= \log 2 \underbrace{\int P_G(x)\, dx}_{1} - \log 2 \underbrace{\int P_{data}(x)\, dx}_{1} \qquad (P_G = P_{data})$$

$V(G,D) = -2\log 2$

$\boxed{V(G,D) = -\log 4}$

when $P_g \neq P_{data}$

to show this, we get rid of assumption that

$P_g = P_{data}$

$$C(G) = V(G,D) = \int P_{data}(x) \log\left[\frac{P_{data}(x)}{P_g + P_{data}}\right] + P_g \log\left[1 - \frac{P_{data}}{P_g + P_{data}}\right] dx$$

$$= \quad " \quad + \underbrace{\left[\log 2 - \log 2 (P_{data} + P_g)\right]}_{0} dx$$

refer slides.

rearranging terms:

$$= -\log 4 + \underbrace{\qquad\qquad}_{\text{Some term}}$$

$= -\log 4 + KL$ div

$= -\log 4 + [\geq 0]$

min attained only when $(-\log 4)$

$C(G) \geq -\log 4$

↓

lower bound

when $P(G) = P_{data}$

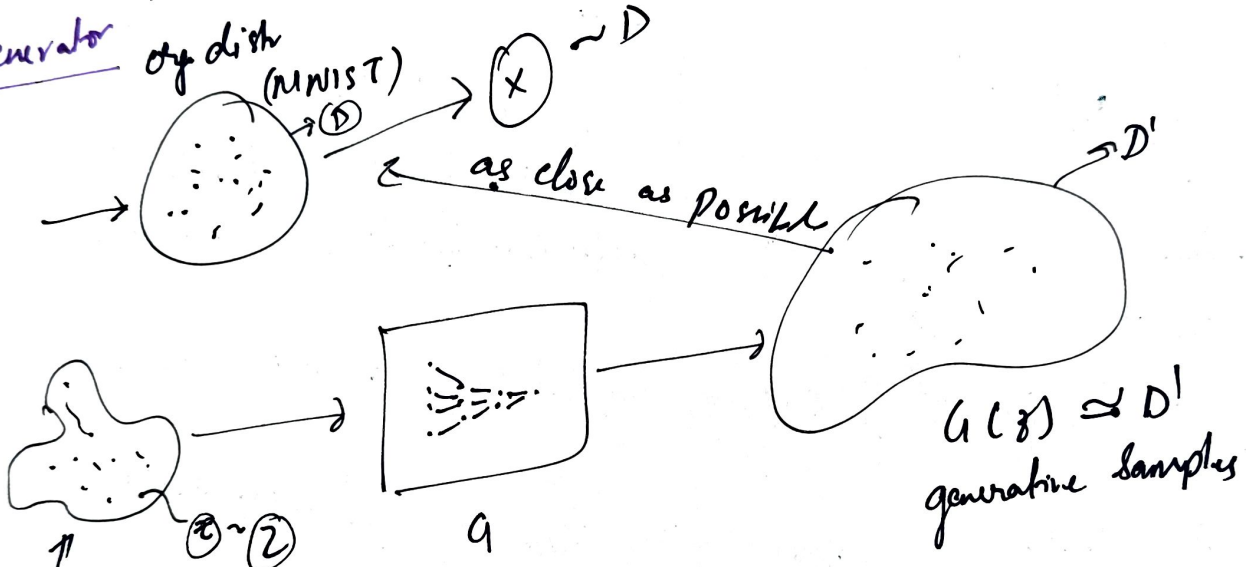
GAN

generator → fight with each other } deep NN / CNN / Vanilla
discriminator
(discriminates b/w real and fake) (0 | 1)
fake real

G trained on x, Sampled on distribution D, Produces D'
(true)

given random distribution z

Generator

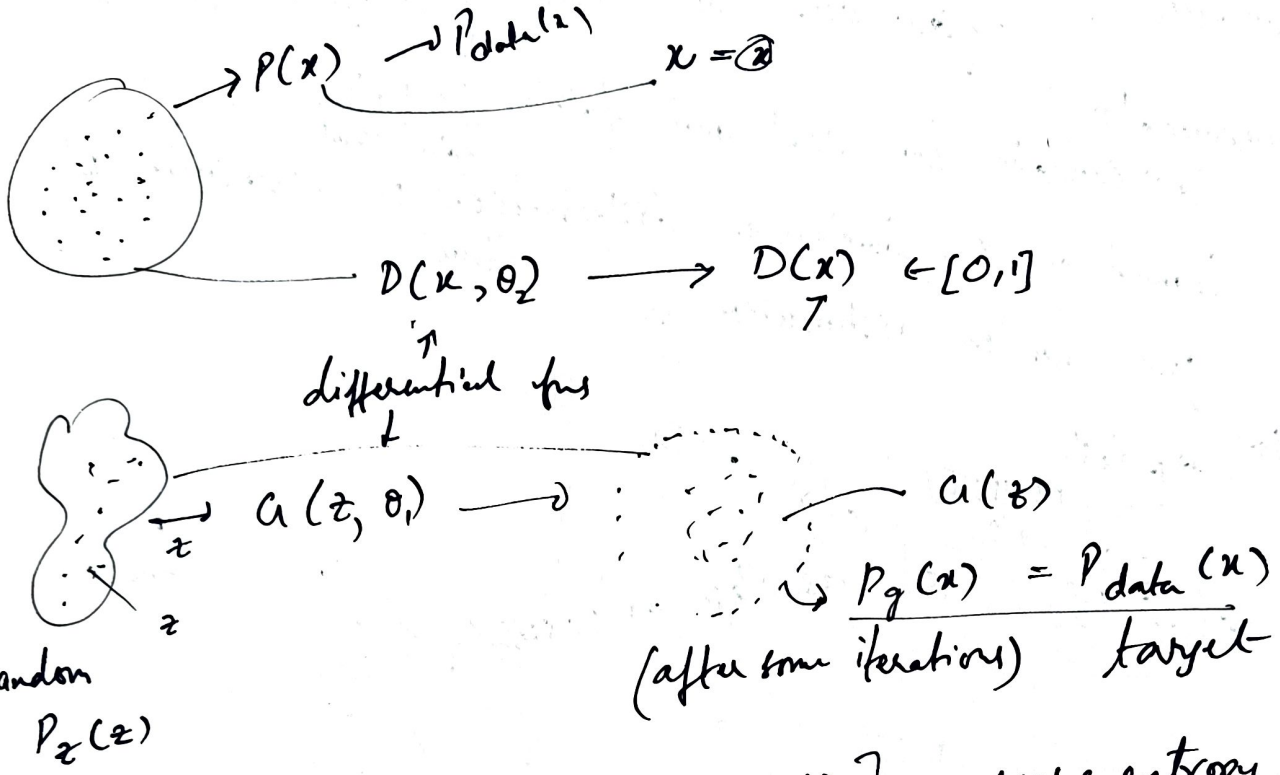


Objective = producing D' which is as close to D.

Optimal Discriminator = 0.5
when it is 0.5, the discriminator fails to differentiate between real and fake data

While training generator, the discriminator is constant, Vice Versa
[easy]
[Chaos]

loss / cost function



$L(\hat{y}, y) = [y \log \hat{y} + (1-y) \log (1-\hat{y})]$ → cross entropy

$\hat{y} = D(x)$ } labels from $P_{data}(x)$ Discriminator
$y = 1$ }

$L(D(x), 1) = \log D(x)$ — (A)

$y = 0$
$\hat{y} = D[G(z)]$ } data from Generator

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

we need (A) and (B) to be maximised
by doing so, (A) will force it $D(x)$ to be 1
and (B) will force $D(G(z))$ to be 0

$$\max \left[\log D(x) + \log (1 - D(G(z))) \right] = D$$

$$\min_G \max_D \left[\log D(x) + \log 1 - D[G(z)] \right]$$

obj fn

Original Eq:-

$$\min_G \max_D V(D,G) = \min_G \max_D \left(E_{x \sim p_{data}(x)} [\log D(x)] + E_{z \sim p_z(z)} [\log 1 - D(G(z))] \right) \#$$

obj of generator = fool the discriminator

$y = 0 \qquad \hat{y} = D(G(z))$

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

$D(G(z)) = 1$ this is possible when we minimize this

Finding best Discriminator

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

derivation

$$D_G^* = \arg\max V(D,G)$$

$$D_G^* = \arg\max \{ E [\log D(x)] + E [\log (1 - D(x))]$$

$E_{P(x)}(x) = \int x \, P_x(x) \, dx$ → we know

GANs END

11:28:33

DIFFUSION MODELS START

11.28.23

Diffusion Models

$$VAE = L_{VAE}(\theta, \phi) = -\log P_\theta(x) + D_{KL}\left[q_\phi(z|x) \,||\, P_\theta(z|x)\right]$$

$$= -E_{z \sim q_\phi(z|x)} \log P_\theta(x|z) + D_{KL}(q_\phi(z|x) \,||\, P_\theta(z))$$

$$\theta^* \; \phi^* = \arg\min L_{VAE}$$

$$-L_{VAE} = \log P_\theta(x) - D_{KL}(q_\phi(z|x) \,||\, P_\theta(z|x)) \leq \log P_\theta(x)$$

~~GAN~~ =

diffusion models → take multiple small steps known as Markov chain

Forward Process

Image → Noise → Noise ---→ Noisy

Reverse Process

Noisy Image → denoise → denoise ---→ Original Image

$$q(x_t|x_{t-1}) = \mathcal{N}\left[x_t ; \sqrt{1-\beta}\, x_{t-1}, \beta_t I\right]$$

$$q(x_{1:T}|x_0) = \prod_{t=1} q(x_t|x_{t-1})$$

Noise - gaussian noise

$$V(G,D) = \int p_{data}(x) \log D(x)\, dx + \int p_z(z) \log(1 - D(G(z)))\, dz$$

diffusion model

$$q(x_t | x_{t-1}) = N\left[x_t, \sqrt{1-\beta_t}\, x_{t-1}, \beta_t I \right]$$

output, mean, variance

$\beta = 0.0001$ to 0.02

to apply this forward formula multiple times:-

$$\alpha_t = 1 - \beta_t$$

$$\bar{\alpha}_t = \prod_{s=1}^{t} a_s$$

hence

$$q(x_t | x_{t-1}) = \sqrt{1-\beta_t}\, x_{t-1} + \sqrt{\beta_t}\, \epsilon$$

$$= N\left[x_t, \sqrt{1-\beta_t}\, x_{t-1}, \beta_t I \right]$$

$$= \sqrt{\alpha_t}\, x_{t-1} + \sqrt{1-\alpha_t}\, \epsilon$$

$$= \sqrt{\alpha_t \alpha_{t-1}}\, x_{t-2} + \sqrt{1-\alpha_t \alpha_{t-1}}\, \epsilon$$

$$= \sqrt{\alpha_t \alpha_{t-1} \alpha_{t-2}}\, x_{t-3} + \sqrt{1-\alpha_t \alpha_{t-1} \alpha_{t-2}}\, \epsilon$$

Calculating $p_\theta(x_0)$ is hard hence we can use Variational lowerbound

$$-\log p_\theta(x_0) \leq -\log(p_\theta(x_0)) + D_{KL}\left[q(x_{1:T}|x_0) \,||\, p_\theta(x_{1:T}|x_0)\right]$$

diffusion models ~ Variational auto encoders

after computing this. we get

$$-\log p_\theta(x_0) \leq \log \frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})}$$ forward proces

$$p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)$$

$$\text{forward} = q(x_{0:T}) = q(x_0) \prod_{t=1}^{T} q(x_t|x_{t-1})$$

$$\text{Reverse} = q(x_{0:T}) = \prod_{t=1} q(x_{t-1}|x_t)\, q(x_T)$$

$$q(x_{t-1}|x_t) = \mathcal{N}\left[x_{t-1} | \tilde{\mu}_t(x_t, t), \Sigma_t\right]$$ Joint distributio

reverse doesnt have closed form, so

$$\tilde{\mu}_t(x_t, t) \approx \frac{1}{\sqrt{\alpha_t}}\left[x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_t\right.$$

DIFFUSION MODELS
END

11.28.23