CS 452
Deep learning Internals

MS : 25
ES : 40
} open notes
Q1, Q2 : 10, 10
Term paper : 10
Demo : 10 (Project)

[Attendence Compulsory]

Absolute grading.

90+ : AB
80+ : BB
60+ : CC
40+ : DD

Two main paradigms of Processing :

1) Algorithmic
   → step-by-step instructions
   → Eg. Processing course registration request, sorting numbers

2) Model based
   → Handle uncertainty, make predictions
   → Eg. choosing your elective, predicting age of a person

Algorithms	Models
Works for every input	They work for certain contexts.
You can always explain how you particular at a result	

* "All models are wrong, but some are useful" - George Box

→ Models simplify reality.
→ we don't know the algorithm
→ we might not know if the algorithm still exist.

Then why we use it?

→ Even if we don't know the true algorithm, models can still be useful.

* Supervised learning

→ we have to gather data
→ we collect data with labels.

Unsupervised learning

→ The model learns on its own.
→ It learns patterns without labels.
→ sometimes the distinction is blurred.
⇒ Its a myth - we always use some form of supervision (supervised learning), directly or indirectly.

Algorithmic solution :-

It might require too much data and too much time consuming.

Model Based examples :-

→ Projectile Motion

→ Relational data model

→ Random Access Machine (RAM) model
- one instruction at a time
each instruction takes equal amount of time

Relational Databases :- used for data storage in tabular form

forceful conversion of bases into rows & columns.

↳ These models help abstract away from low-level details and focus on patterns

---

Statistical Models :-

Why do we build stastical models ?

↳ Built when we can't write rules explicitly.

↳ Try to model underlying patterns from the data.

Explainable vs Non-Explainable Models :-

* Model Parameters :
(statistical Models requires various Parameters)

→ Find the appropriate value of the Parameters (like weights in ML).

→ Process :
- → Initial Guess
- → Predict output
- → compare with actual output (calculate Error)
- → Update guess (using learning algorithms).

Example :-

Input	output
1	2
2	6
3	10
10	38

shows a pattern in data that we want a model to learn.

- linear Model
  $y = ax + b$
- Parameters : a, b

make Initial guess : a = 1, b = 0,
use a cost function (like MSE)

to measure error.

- Cost function: To predict how better the guess is.
- Gradient descent to update a & b.

↳ we need to reduce cost function but it is not the only thing.

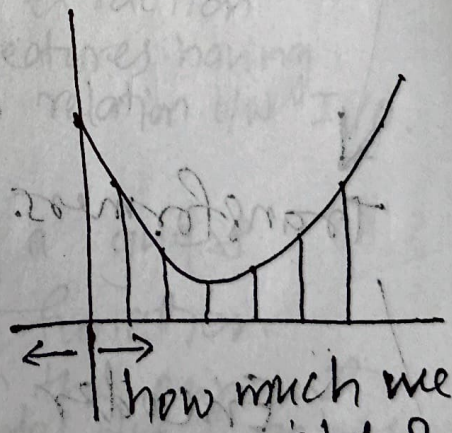
~~(It's like an exam)~~ ~~exam ☹~~

~~real world pred~~

exam performance vs Real world Performance.

- Gradient Descent
  - stochastic Gradient Descent (SGD)
  - learning rate
  - Extending to an ANN.
- Types of Gradient Descent
  - → Batch
  - → SGD.

* learning rate
  - → Controls how much we move in the direction of min error.
  - → Too small: slow learning
  - → Too large: may overshoot the optimum



how much we go right?

learning rate? (we will take baby steps).

Overview (Course)

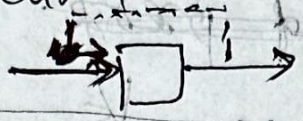
Model Based Solutions (Paradigms)

Feed-forward v. Neural Network

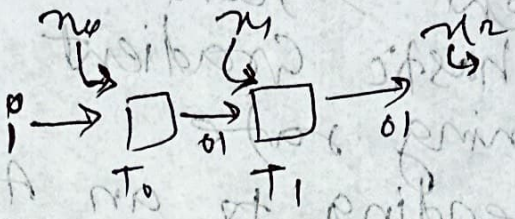
Artificial

Backpropagation

Recurrent Neural Network [Vanilla, LSTM GRU]



Attention mechanisms



RNN Configuration

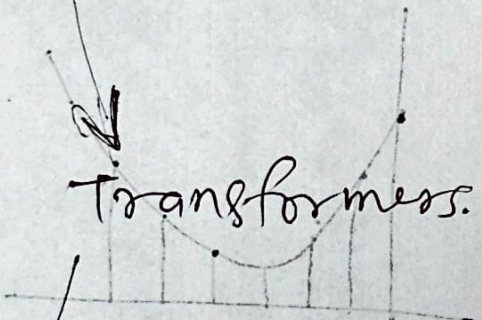
History (Recurrent)

How to selectively use the history (Attention Mech.)

Transformers

RNN — ~~Recurrence~~ Recurrance

Keep Attention



Encoder / Decoder

① Encoder only (BERT)

② Decoder only (LLMs)

Hardware (CPU, GPU)

various Open problems / Issues

Inference, Fine Tunning, Bias, Ethics

→ Problems where algos can't be used

① No algos ~~can't~~ is possible

② Algo. might exist, not sure.

③ Algo. exists, but

i) Inefficient (Uses too much resources, time, memory, network bandwidth)

ii) We want a diff. insight

∴ We abandon Algo. & uses model based solutions.

→ Blackbox

Input	Output
⋮	⋮
⋮	⋮
⋮	⋮

large training dataset

from input/output guess the logic behind the relation.

traditional way:
features extraction
find features having high relation b/w I/O.

f(d)
↓
features
Embeddings

① Make a guess

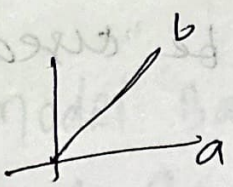
② cost function (with the current model did I get the same output)

How bad the model is

③ Change the parameters to reduce the error (cost function)

eg: $\frac{1}{2}\sum_{i=1}^{|D|}(ti - Oi)^2$

for, distance checking



$-\eta \frac{\partial E}{\partial a}$ learning rate

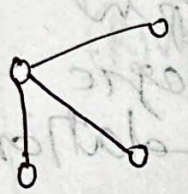
$a = a + \Delta a$

$\Delta a \rightarrow -\eta \frac{\partial E}{\partial a}$

learning rate required so that we don't change the parameters abruptly, slow changes are required.

ANNs :—

Neurons in brains $\leftrightarrow$ Nodes in graph connected to other nodes.



$\frac{10^{11} \times 10^{4}}{2} = \frac{10^{15}}{2} = 5 \times 10^{14}$

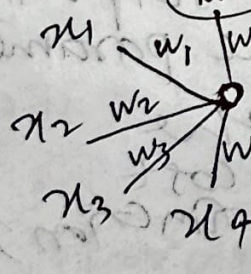
parameters in human brain

10^3 switching time human brain

10^9 switching time for computer models.

so that if all inputs are zero can we still get an output

⇒ Perceptron :—



Pass : 1
fail : -1
→ marks of student

$X = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$ ↓ Inputs

$W = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix}$ ↓ Parameters

Thresholded perception = (+1, -1 O/P)

$\vec{x} \cdot \vec{w} > 0$ then o/p = 1
$f(\vec{x} \cdot \vec{w})$ else o/p = −1 } → output of a perceptron

Training (earlier days): $w_i = w_i + \Delta w_i$, $\Delta w_i = \eta (t - o) x_i$

↳ Guarantees convergence if data is linearly separable.

What if data is not linearly separable?

→ Delta rule
→ Gradient descent
→ Dot product of weighted vector & input vector
→ Unthresholded perceptron: $o(\vec{x}) = \vec{w} \cdot \vec{x}$
→ cost function (error function) $E(\vec{w}) = \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$
→ What to do → w_i ← Inc. → How much, const., dec. → How much; $w_1 \leftarrow$, $w_2 \leftarrow$ ....
↳ Gradient of cost function $\nabla E = \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots \frac{\partial E}{\partial w_n} \right]$

$w_i = w_i + \Delta w_i$
↳ ~~to find~~ ~~Eu~~
$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$

$\frac{\partial E}{\partial w_i} = \frac{1}{2} \sum_{d \in D} \frac{\partial (t_d - o_d)^2}{\partial w_i}$ → $2 (t_d - o_d) \cdot \frac{\partial (t_d - o_d)}{\partial w_i}$

$\frac{\partial (t_d - o_d)}{\partial w_i} = -\frac{\partial o_d}{\partial w_i}$

$-\frac{\partial (\vec{w} \cdot \vec{x}_d)}{\partial w_i}$ ← output $o_d = \vec{w} \cdot \vec{x}_d$

$\vec{w} \cdot \vec{x_d} = w_0 x_0 + w_1 \cdot x_1 + w_2 x_2 + w_3 x_3 + w_4 \cdot x_4$

$$\frac{\partial w_i x_i}{\partial w_i} = x_i$$

$$\Rightarrow -\frac{\partial O_d}{\partial w_i} = x_i$$

$$w_i = w_i + \Delta w_i$$

$$\downarrow$$

$$-\frac{\eta \partial E}{\partial w_i} = +\eta \left[\frac{1}{2} \cdot 2(t_d - O_d)\left(+\frac{\partial O_d}{\partial w_i}\right)\right]$$

$$\Delta w_i = \underbrace{\eta (t_d - O_d) x_i}$$

similar to the Δw_i of earlier days (i.e similiar for linearly separable)

E is also these

η (+ve) $(t_d - O_d)$ x_i (+ve)

$(t_d - O_d)$: +ve / −ve

Δw_i +ve $\Rightarrow w_i \uparrow$

Δw_i −ve $\Rightarrow w_i \downarrow$

O_d may differ, η, t_d, x_i remains same for a w_i

But here output is different (previously only two outputs +1, −1)

$w_0^0 \rightarrow w_0^1 \rightarrow w_0^2 \rightarrow w_0^3 \rightarrow \cdots$

stop when the updates are not significant or when resources ends.

Pick one research paper
Published in 2024-2025

Term Papers

from → Conference
- ACL ☆
- NewIPs ☆
- NAACL ☆
- AACL ☆
- AAAI
- EMNLP ☆
- CVPR
- ICLR

CS

Journal
Nature
Science
Cell

Preprints
arXiv
But no quality check ☆

CORE
↓
funding agency for CS research papers
→ Ratings
- A+
- A
- B
- C
- other

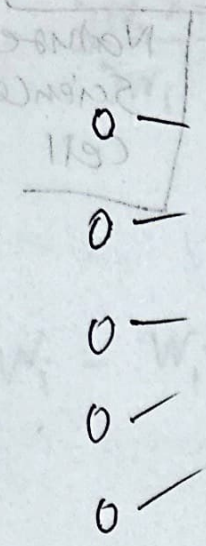
Presentation & Demo

(Explain the paper) → Before mid-sem

Demo → After Midsem

Research Paper
→ Title
→ Abstract
→ Intro.
→ our work
→ Experiments
→ Related work
→ Conclusion
→ Limitations

instead of taking one neuron take multiple neurons.



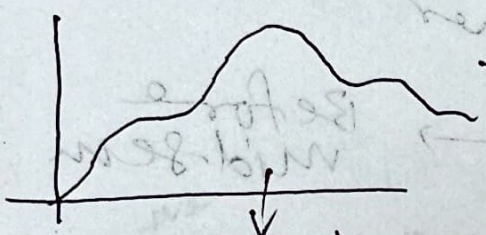
Similar to, $y = a_1 x + b_1$

$a_2 x + b_2$

$a_3 x + b_3$

multiple linear functions will still give linear function thats a problem

∴ compute non-linear function

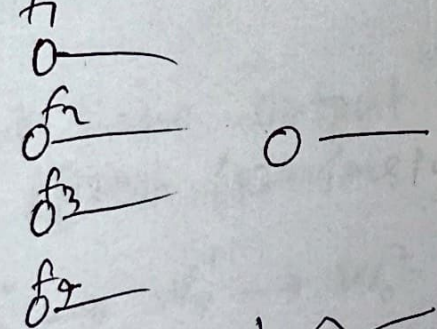


linear functions can't give this type of non-linear outputs

∴ use non-linear functions.

output required (from data)

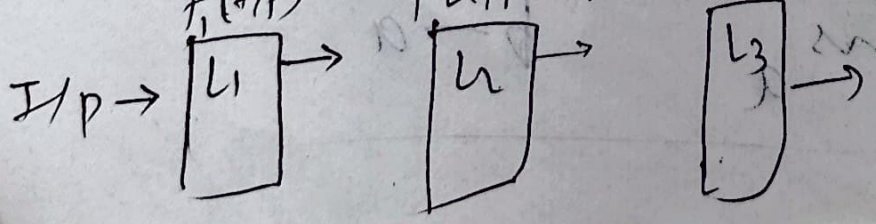
$f_1, f_2, \ldots$ are non-linear functions

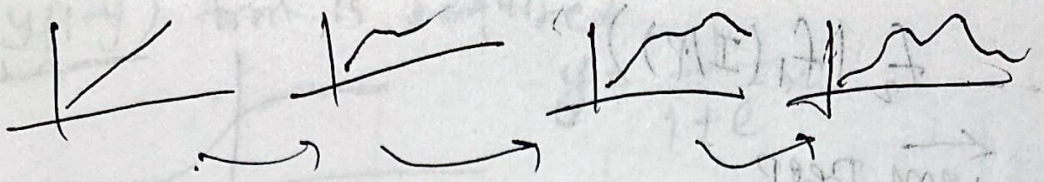


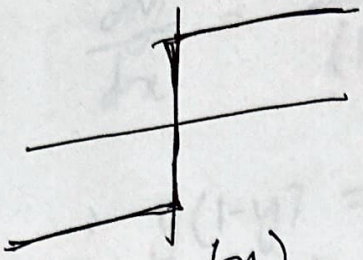
use multiple layers & neurons to get the output as close as the desired output.

layer 1 layer 2

$f_1(I/P)$ $f_2(f_1(I/P))$ $f_3 f_2 f_1 (I/P)$





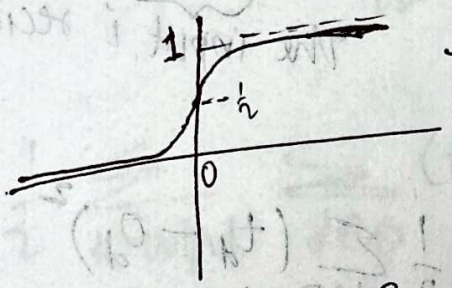


→ non differentiable
∴ its a problem

Sgn(x) → used in earliest models (perceptron threshold)

Signum function

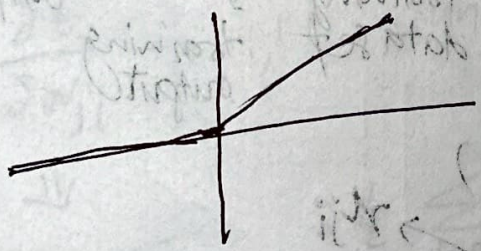
∴ Use sigmond funct. $f(x) = \frac{1}{1+e^{-x}}$

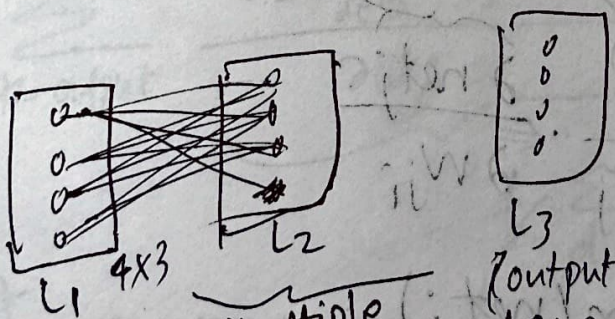


→ Its differentiable so we can use GD, gradient descent, etc

⇒ Activation function

Or RELU (Rectified ___ function)





L1 (Input layer)

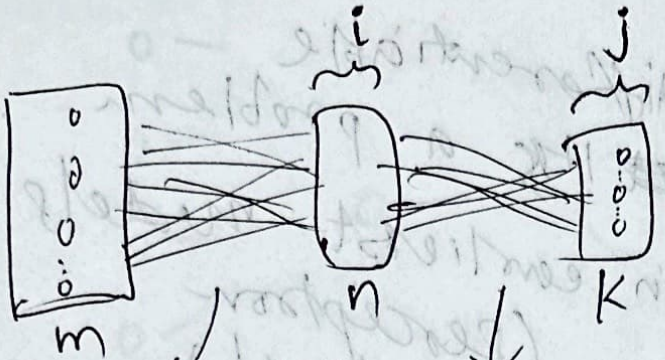
Multiple Hidden layers

L3 (output layer)

completely connected (all neurons of one layer connected to all the neurons of next layer)

$f_{100} \cdots f_{200} \cdots f_2(f_1(I/P))$

← very Deep simple functions



$m \times n + n$ weights ← bais fake input

$n \times K + K$ weights

→ bias fake input to get non zero value

net input recieved at neuron j $\Rightarrow net_j$

weight w_{ji}

output o_j

x_{ji} the input i recieved

$$w_{ji} = w_{ji} + \Delta w_{ji}$$

$$-\eta \frac{\partial E}{\partial w_{ji}} \rightarrow \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

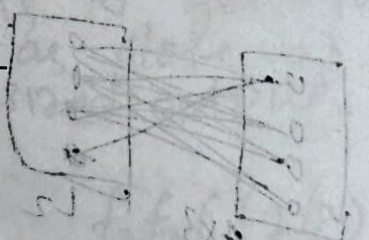
$d \in D$: training data set; t_d: training output; o_d: model output

$$\frac{\partial E}{\partial o_j} \cdot \frac{\partial o_j}{\partial w_{ji}}$$

$$\frac{\partial o_j}{\partial net_j} \cdot \frac{\partial net_j}{\partial w_{ji}}$$

$\frac{\partial o_j}{\partial net_j} = o_j(1-o_j)$

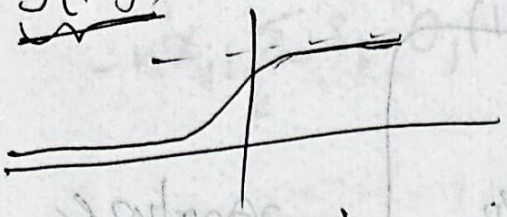
$\frac{\partial net_j}{\partial w_{ji}} = x_{ji}$



$o_j = activation(net_j)$

required Δw_{ji} R

$y(1-y)$ form is required



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

$$\frac{dy}{dx} = \frac{-1}{(1+e^{-x})^2} \cdot \frac{d(1+e^{-x})}{dx} = \frac{e^{-x}}{(1+e^{-x})^2}$$

$$\therefore y(1-y) = y - y^2 = \frac{1+e^{-x}}{(1+e^{-x})^2} - \frac{1}{(1+e^{-x})}$$

$$\frac{\partial O_j}{\partial net_j} = O_j \cdot (1 - O_j)$$

$$net_j = \sum_{\substack{\text{all} \\ \text{dimensions} \\ m}} x_m w_m \rightarrow \frac{\partial net_j}{\partial w_{ji}} = x_{ji}$$

$$\frac{1}{2} \sum_{k=\text{output}} \sum_{d \in D} (t_{dk} - O_{dk})^2$$

↳ go through all neurons and the training dataset

$$\frac{\partial E}{\partial w_{ji}} \Rightarrow \sum_{k=\text{output}} \frac{\partial \frac{1}{2} \sum_{d \in D} (t_{dj} - O_{dj})^2}{\partial O_j}$$

$$\Rightarrow \frac{1}{2} \sum_d 2(t_{dj} - O_{dj}) \cdot \underbrace{\frac{\partial (t_{dj} - O_{dj})}{\partial O_j}}_{-1}$$

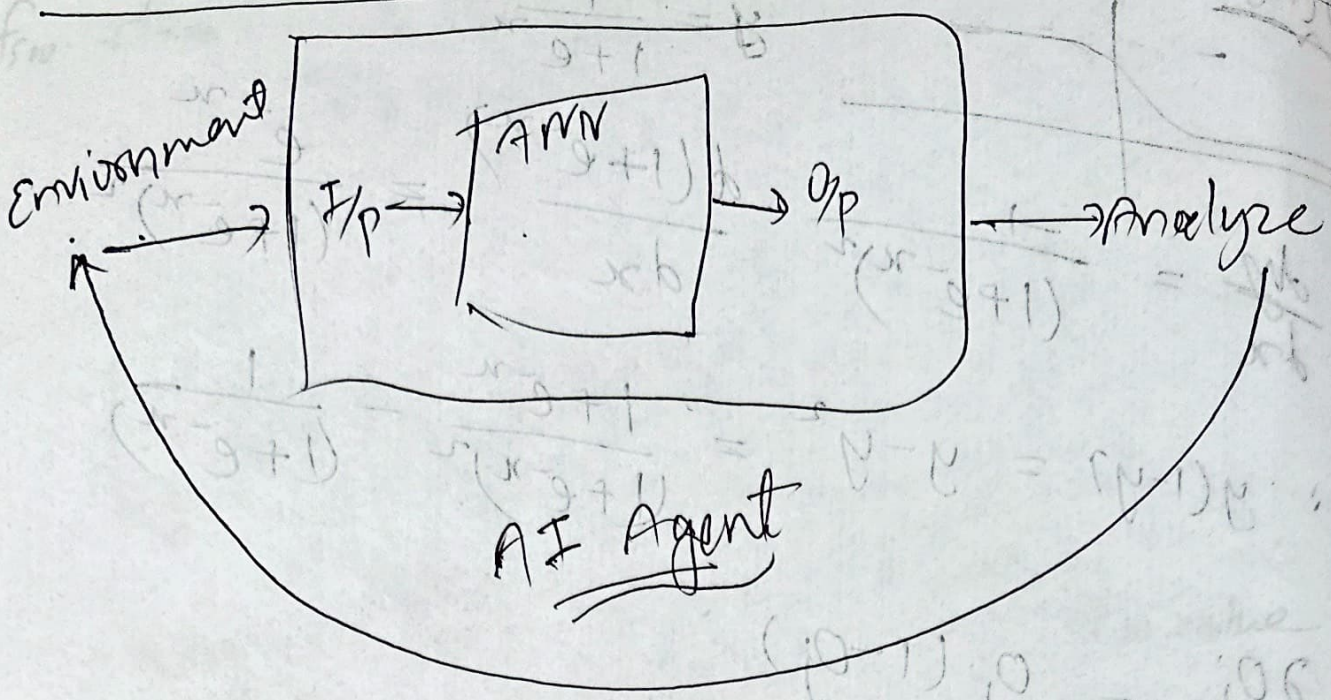
-ve of derivative of error wrt. net j $\Rightarrow \frac{\partial E}{\partial net_j}$

Backpropagation

$$\Delta w_{ji} = -\eta \cdot \delta_j \cdot x_{ji}$$

η → learning rate

δ_j → For output layer



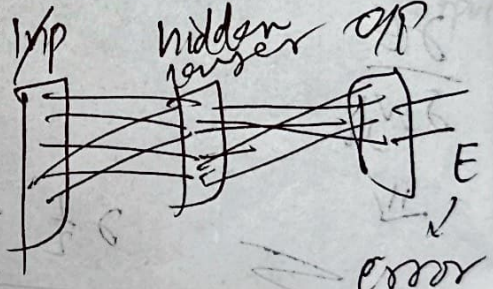
① Frozen LLM Agent

② RL Agent

$\Delta w_{ji} = \eta \delta_j x_{ji}$ → for output layer.

$\delta_j = \frac{-\partial E}{\partial net_j}$

⇒ for hidden layer



$\Rightarrow -\eta \frac{\partial E}{\partial w_{ji}}$

$\frac{\partial E}{\partial net_j}$ (= δ_j) $\cdot \frac{\partial net_j}{\partial w_{ji}}$ ($\rightarrow x_{ji}$)

$\frac{\partial E}{\partial net_j} = \sum_{k \in Downstream(j)} \frac{\partial E}{\partial net_k} \cdot \frac{\partial net_k}{\partial net_j}$ ($= \delta_k$)

$\frac{\partial net_k}{\partial net_j} \rightarrow \frac{\partial net_k}{\partial O_j} \cdot \frac{\partial O_j}{\partial net_j}$

$\frac{\partial net_k}{\partial O_j}$ ($= w_{kj}$)

$\frac{\partial O_j}{\partial net_j}$ ($= O_j(1-O_j)$)

$-\eta \alpha_{ji} \sum_k \delta_k \cdot O_j(1-O_j) W_{kj}$ → For hidden layer

---

Word To Vec :-

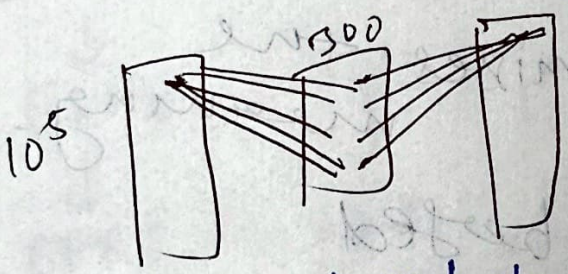
Fake Task

predict the neighbour words.

I went school today

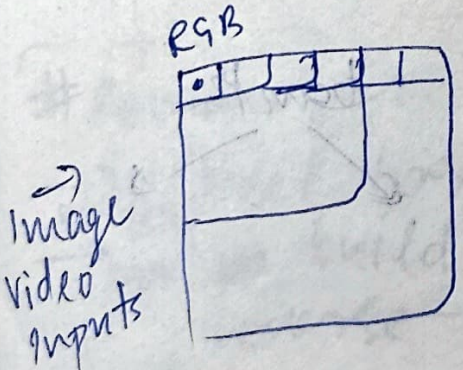
predict the missing word.

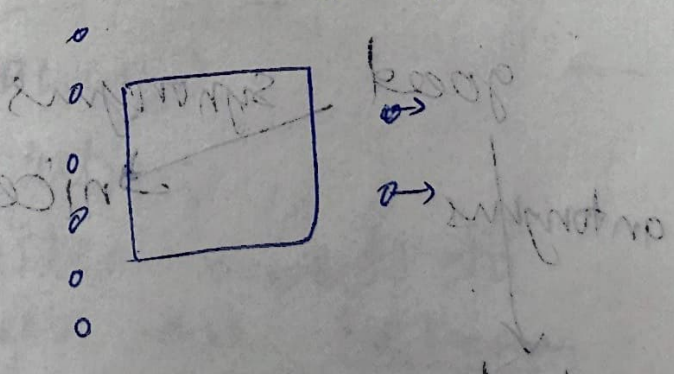
↳ static representation.



How to give Input to ANN ??

Data - Nos, text, images, audio, time series, graphs, etc.





→ waveforms

time domain
freq. domain

↳ Audio inputs

Graphs can be given as input as Adjacency lists or nxn matrix.

How to give text as inputs.

Earlier way → One hot representation

a

d_1 [1]
d_2 [0]
[0]
[:]
d_{10^5} [0]

apple

[0]
[0]
[0]
[1]
[0]
[0]
[:]
[0]

Problems with one hot representation:
It completely misses the meaning,
New words comes up with time

2nd way → ASCII representation
→ represent words at character level.
→ All words don't have same length, it also misses the meaning.

3rd way → Rule based
↓
Word Net

good
- synonyms → nice
- antonyms → bad

bank

limitations of symbolic representation

- Related words
- New words
- Change of meaning
- Multiple meaning

→ Distributed representation,
Distributional semantics
Words are represented
in the form of vectors.

Back prop.

↓

Word 2 Vec 2013

↓

D2 Vec

↓

ELMo 2018

↓

BERT 2018

↓

LLMs 2019 –

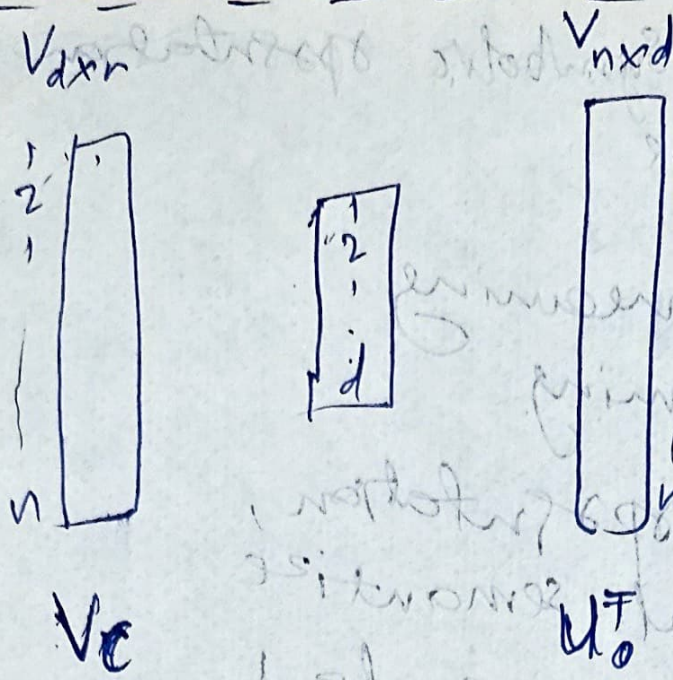
#Remember: (Takeaways)

→ Model based solutions
we need

→ We build statistical models to reverse engineer the system

I/P → O/P

→ Linear regression
↳ make a guess → evaluate → improve



$$E(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{j}\log\left(P(W_{t+j}\mid W_t)\right)$$

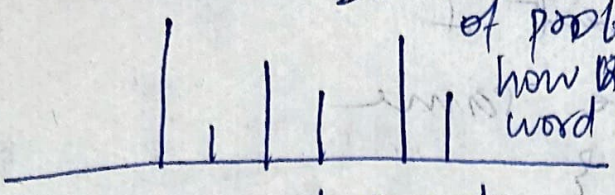
for evaluation we have loss function
for improvement we have any optimization process
i.e, Gradient Descent

↓ Backprop. ↗ can't use GD

→ Perceptron {Non-linear output}
→ Unthresholded perceptron {linear}
→ Differentiable Neuron
Activation function → Segmoid
→ How to represent input → Distributional semantics
↖ Distributed representation

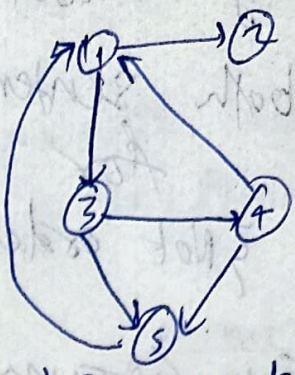
10% weightage — Syllabus till last friday — open Notes

Quiz on Thursday → during class time



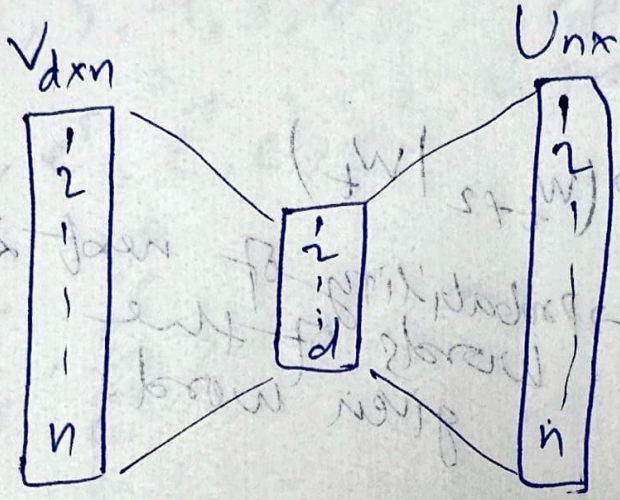
→ Distribution of probability of how often a word is used.

apple : fruit, red, sweet, ... etc
: company, steve jobs, etc



webpage ranking

Word to Vec :-



I/P
↑
each word is a column

O/P
↑
each word is a row

skip Gram model :-

I went [to] school today

{predict the 2 words back & front of the given word}

CBOW {Cont. bag of words} model.

I went [to] school today

{Guess the missing word}

Today school to went I

{both sentences are same for CBoW}

{Not order dependent}

Skip Gram Model:-

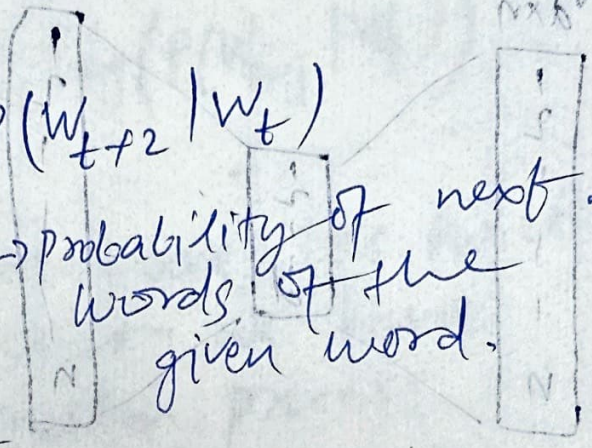
$P(W_{t+1} | W_t)$

W_{t+1} ↓ next word, W_t ↓ given word

Probability

$P(W_{t+1} | W_t) \cdot P(W_{t+2} | W_t)$

↳ probability of next 2 words of the given word.



↓ Independent events

→ For one token

$$\prod_{\substack{j=-2 \\ j \neq 0}}^{2} P(W_{t+j} | W_t)$$

↳ probability of surrounding words at distance 2.

$$\sum_{\substack{j=-2 \\ j \neq 0}}^{2} \log\left(P(W_{t+j} | W_t)\right)$$

→ more value means the model is How good

⇒ take -ve of this gives the cost function {How bad the model is}

for All tokens {Eg. words}

$$\Rightarrow -\sum_{t=1}^{T} \sum_{\substack{j=-2 \\ j \neq 0}}^{2} \log(P(w_{t+j} | w_t)) \rightarrow \text{cost function}$$

↳ window size of 2.
{Hyper Parameter}

Softmax

2, 5, -1, 7, 0

↓ ↙ Softmax Transformation

$e^2, e^5, e^{-1}, e^7, e^0$

↓

$\frac{e^2}{\Sigma}, \frac{e^5}{\Sigma}, \dots = 1$

Resource
Parameter learning in word2vec

Christopher Manning Stanford lectures