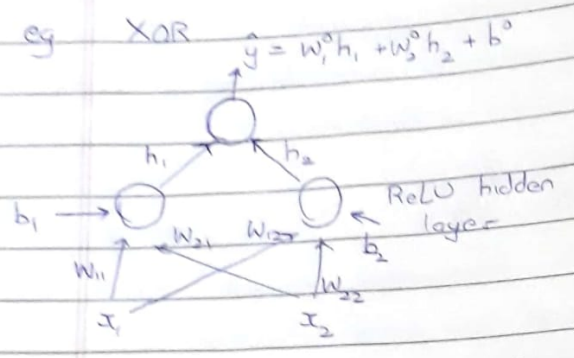
Consider the case for 'square

eg XOR



Initially, all $W = 0$

$b^0 = 0, \; b_1 = 1, \; b_2 =$

One iteration : over one training example $(x_1 \; x_2 \; y) \equiv (1 \; 0 \; 1)$

Step 1 Forward Pass

$h_1 = ReLU(W_{11} x_1 + W_{21} x_2 + b_1) = \max(0, 1) = 1$

$h_2 = W_{12} \quad W_{22} \quad b_2 = -1 = 0$

$\hat{y} = w_1^0 h_1 + w_2^0 h_2 + b^0 = 0$

Square loss $= L = (y - \hat{y})^2 = 1$

Step 2 Back Propagation

$$L = (y - \hat{y})^2$$

1 $\dfrac{\partial L}{\partial \hat{y}} = -2$

$\dfrac{\partial L}{\partial w_1^0} = \dfrac{\partial L}{\partial \hat{y}} \times \dfrac{\partial \hat{y}}{\partial w_1^0} = -2 h_1 = -2$

Update w_1^0, w_2^0 similarly & b^0

2 $\dfrac{\partial L}{\partial h_1} = \dfrac{\partial L}{\partial \hat{y}} \times \dfrac{\partial \hat{y}}{\partial h_1} = -2 w_1^0 = 0$

$\dfrac{\partial L}{\partial W_{11}} = \dfrac{\partial L}{\partial h_1} \times \dfrac{\partial h_1}{\partial W_{11}} = 0 \cdot \dfrac{\partial ReLU(W_{11} x_1 + W_{21} x_2 + b_1)}{\partial W_{11}} = 0 \cdot 1 \cdot x$

$= 0$

Similarly update W_{11} & W_{21} & b_1

F] Regularization

- Reducing generalization error of a model, even at the c
  of training error

• Objective function = loss + α regularize

A Regularization :- Early Stopping

- Training error always decreases, validation error decreases, t
  increases in case of overfitting
- Stop training when validation set increases more than
  certain number of times

RECURRENT NEURAL NETWORKS

→ For sequential inputs $x_1 \dots x_n \rightarrow y$

1 Classifying sequences
eg - Sentiment classification

2 Next word in a sequence $x_1 \dots x_n \rightarrow x_{n+1}$
eg - Language modeling, predictive text

3 Label per token in a sequence $x_1 \dots x_n \rightarrow y_1 \dots y_n$
eg - Parts of speech

4 Sequence prediction $x \rightarrow y_1 \dots y_m$
eg - Translation

- Examples :-

eg Forecasting (in a time series)

I/p :- $x_1 \dots x_{at}$
where x_i = day of the week/holiday or not/temperature

O/p :- $y_1 \dots y_t$
where y_i = demand for an item

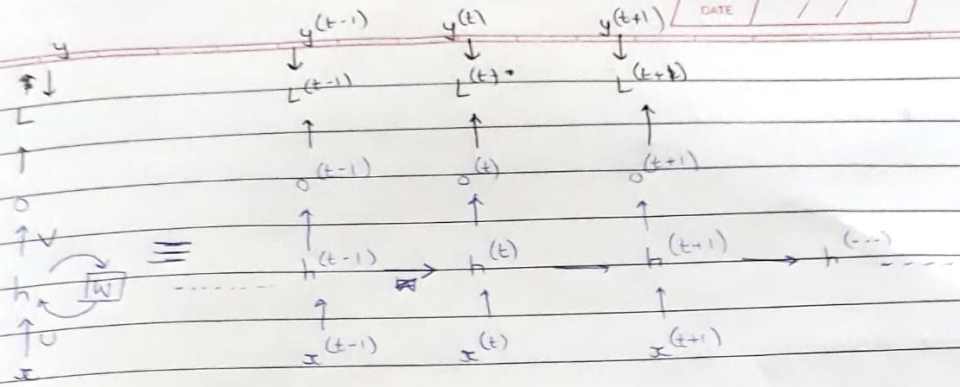
I A RNN

- Processing ID input of variable length

* In CNN, each hidden output is a function of corresponding i and some immediate neighbours.

- In RNN, each output is a function of a 'state' summarizing previous inputs and current input.
  - State summary is computed recursively.

- RNN allows deeper, longer-range interaction among parameters than CNNs for the same cost.



$o^t = c + Vh^t$

$\hookrightarrow h^t = \sigma(b + Wh^{t-1} + Ux^t)$

↑ Sigmoid (convention)

* The end of sequence is represented by a 'token' in the input

• Parameters :- U, V, W

→ RNN :- Forward Computation Example

Sequence $(x_1, y_1) \dots (x_t, y_t)$

where $x_i \& y_i \in \mathbb{R}$

Say $h \in \mathbb{R}^4$, $\Rightarrow W \in \mathbb{R}^{4\times4}$, $V \in \mathbb{R}^{1\times4}$, $U \in \mathbb{R}^{4\times1}$

$b \in \mathbb{R}^{4\times1}$, $c \in \mathbb{R}^{1}$

Consider $h^0 = 0$

Calculate $h^1 = \sigma(Wh^0 + Ux_1 + b)$

$o^1 = Vh^1 + c$

Consider square loss :- $L_1 = (y_1 - o^1)^2$

Similarly find h^2, o^2, L_2

* Practically, we actually lengthen input sequence ('padding') by junk data. While calculating loss, we must mask the padding contribution

A] → RNN for text (Word Prediction)

- We 'embed' ~~as~~ distinct words to the real number space

Every x_t denotes a word $\in [1 \dots V]$

V = 'Vocabulary Size' ≈ 30000

For each of the 30000 words, we make a vector embedding

Embedding matrix $S_{V \times d}$

where $d \sim 300$

Define ~~S_{x_t}~~ $S_{x_t} = x_t S$ = vector embedding of that word

x_t is a $1 \times V$ one-hot vector

(1 at one position depending on word input, 0 at all other elements)

We now give $S_{x_t} = x_t S$ as input to RNN to compute correct h^t

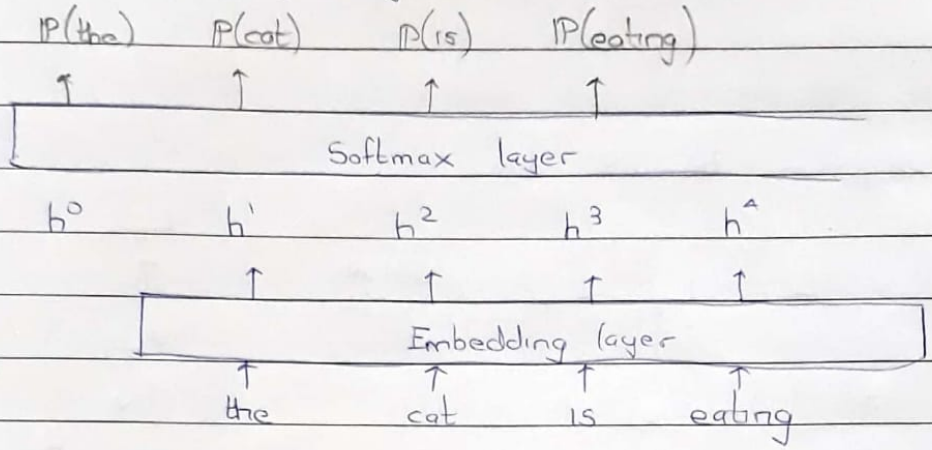
* Words of similar meaning are placed closer in the (standardized) 300-dimensional space

$S \rightarrow$ S is a look-up table that was learned

→ Training a sequence model

- Maximum Likelihood :- $\mathbb{P}(y \mid x, \theta) = \prod \mathbb{P}(y_t \mid y_1 \cdots y_{t-1}, x_t, \theta)$

eg $\mathbb{P}$ (the, cat, is, eating)



- Training data

$D \equiv \{(x^1, y^1) \cdots (x^N, y^N)\}$

Language model :- We want $y^{t-1} = x^{t-a}$ (prediction)

Training data :- $\vec{x}$:- The cat is ~~sleep~~ eating

$\vec{y}$:- cat is eating EOS

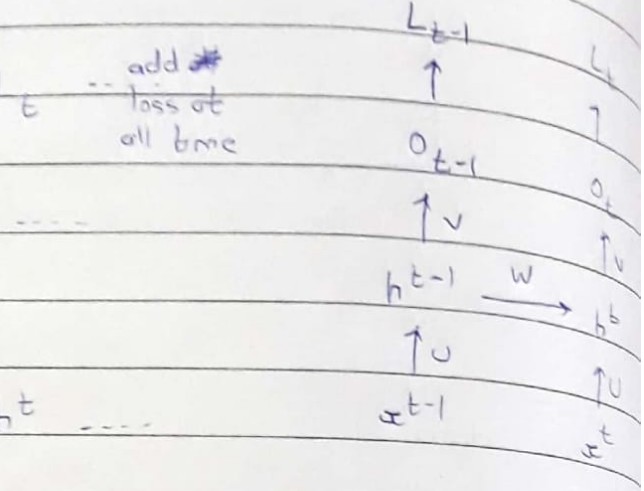
↳ end-of-sequence.

See following example

A

→ Backpropagation through time

Total loss = $L = L_{t-1} + L_t$ -- add all loss at all time



$$-\frac{\partial L}{\partial V} = \frac{\partial L_{t-1}}{\partial V} + \frac{\partial L_t}{\partial V} \quad \cdots$$

$$= \frac{\partial L_{t-1}}{\partial O_{t-1}} \frac{\partial h^{t-1}}{} + \frac{\partial L_t}{\partial O_t} h^t \quad \cdots$$

$$-\frac{\partial L}{\partial W} = \frac{\partial L_{t-1}}{\partial W} + \frac{\partial L_t}{\partial W}$$

$$= \frac{\partial L_{t-1}}{\partial O_{t-1}} \times \frac{\partial O_{t-1}}{\partial h_{t-1}} \times \frac{\partial h^{t-1}}{\partial W} + \cdots$$

$$= \frac{\partial L_{t-1}}{\partial O_{t-1}} \times V\sigma'\left(h^{t-2} + W\frac{\partial h^{t-2}}{\partial W}\right) + \cdots$$

This will have all terms till $\frac{\partial h^0}{\partial W}$ with coefficient W^t (can explode or vanish)

☹ We have to go back over all h^n to compute gradient

'Exploding or Vanishing Gradient'

Gradient is either small or large

PAGE NO.
DATE / /

- Solutions for vanishing/exploding gradient

Multiple time scales :-
Add direct connection from far past inputs to output instead of depending on state to capture all past inputs.

∴ Cannot change how far backs we look at different times or for different inputs.

Solution :- Gated RNNs, (eg - LSTMs)

3/10

B] ⟶ Sequence Prediction

$\vec{x} = (x_1 \cdots x_n) \longrightarrow \vec{y} = (y_1 \cdots y_n)$

eg - sentences, image, audio

Every y_i is called a 'token' (any term from a huge vocabulary)

eg Translation :- <English sentence> ⟶ <Hindi sentence>

eg Image captioning :- <Image> ⟶ <Describing sentence>

eg Conversation assistance

eg Speech recognition :- <Speech spectrogram> ⟶ <Phoneme sequence>

- Challenges

1) Long range dependency
   - Does not assume conditional independence
   eg Ram eats. He also sings

2) Highly open-ended prediction space
   - Even length of output is variable

When $\arg\max_y P(y \mid \ldots)$ = stop the sentence

A Encoder - Decoder Model

1 Encode input elements into a fixed-dimension vector X. (embedding)

2 Decode $\vec{y}$ token by token using an RNN
   - Initialize RNN with state X.
   - Repeat until RNN generates an EOS token :-
      - Feed previously generated token as input
      - Get a distribution over output tokens and choose the best

Write $P(y_t \mid y^{(t-1)}, X, \theta) = P(y_t \mid h_t, \theta)$

h_t: state vector implemented by RNN

with $h_0 = X$

$\arg\max_{y_t} P(y_t \mid h_t, \theta)$ is a word known as 'token'

Stop RNN if token = EOS

PAGE No.
DATE / /

- We are doing greedy search :- Word prediction #n does not depend on word #k for $k \geq n$

→ Attention-based sequence learning :-

↳ Alignment b/w input sequence & output sequence
(ex correspondence b/w English words and Hindi words)

- An RNN can attend to ~~inputs~~ output of another RNN.

Logit = 'Attention' = $A_{tj} = z_{t-1} \cdot h_j$ (dot product)

Take Softmax (A_{tj}) for producing t_j^{th} output element.

'Attention-weighted input states'

$\Sigma a_{tj} h_j$ to produce z_t

A30
600

CLUSTERING

Unsupervised Learning

- Data is unlabelled. We want to group similar elements.

Input :- $D \equiv \{x^1 \ldots x^N\}$

1 Dimensional Representation :-
Let $x^i \in \mathbb{R}^d$. Find Euclidean distance

OR

2 Distance function
$d(x^i, x^j) \in \mathbb{R}$

→ Parameters
1 $k \equiv$ No. of clusters (taken as input)
2 Hierarchy of clusters
3 Correlation clustering :- 'signed' distances

* Deduplication :- Remove duplicate addresses
If instances belong to same address → +ve distance
different → -ve

- Based on clusters :-
Interested in :-
1) Centroid of each cluster
2) Partitioning data into clusters

→ Objectives :-
1 Assign one of k clusters $C_i(i)$ to each instance 'i'

$C_1 \cup C_2 \ldots C_k = D$
$C_i \cap C_j = \emptyset$

PAGE NO.
DATE / /

Minimize $\sum_{k=1}^{k} \left(\sum_{(i,j) \in C_k} d(x^i, x^j) \right)$ .... pairwise distances in each cluster.

2. Cluster centers $m_1 \dots m_k$

- Most popular distance function ≡ Euclidean

Algorithm Begin at $t = 0$ K-MEANS ALGORITHM

$m_1^t \dots m_k^t$ ≡ random k points from $X^1 \dots X^N$

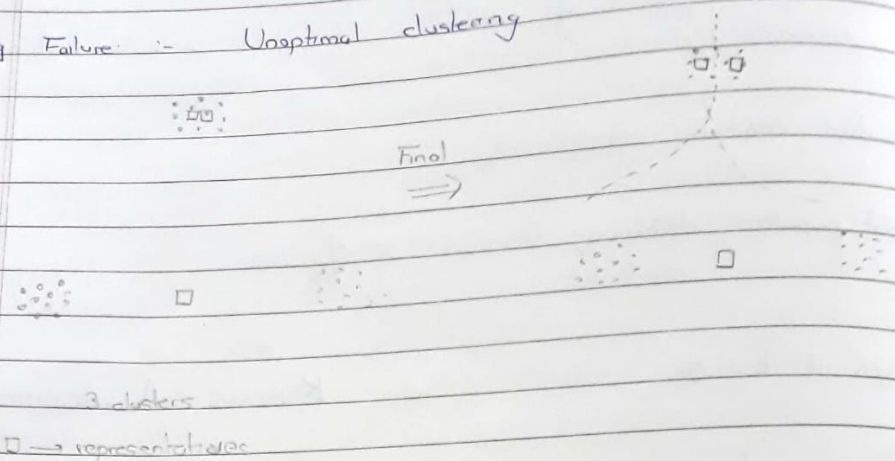
↳ // representative of each cluster

$c^t(x^i)$ → randomly from $1 \dots k$

```
while (changing) {
    for (i = 1 to N) {
        c^{t+1}(x^i) = argmin_{k ∈ {1...k}} d(x^i, m_k^t)
    }
    for (k = 1 to k) { // update representatives
        m_k^{t+1} = min_m Σ_{x^i in C_i} d(x^i, m)

        // If d is Euclidean, then m_k^{t+1} will be equal
            to mean of all points in that cluster
                (Central Limit Theorem)
    }
}
```

eg Failure :- Unoptimal clustering



- Solution :- Run multiple times, with different initial clustering

→ Proof of convergence of K-means algorithm

Define $F(\{c(x^i)\}, \{m_k\}) = \sum_{k=1}^{k} \sum_{\substack{i=1 \\ c(x^i)=k}}^{N} d(x^i, m_k)$

Step 1 :- By design,

$$F(\{c^{t+1}\}, \{m_k^t\}) \leq F(\{c^t\}, \{m_k^t\})$$

Step 2 :- By design,

$$F(\{c^{t+1}\}, \{m_k^{t+1}\}) \leq F(\{c^{t+1}\}, \{m_k^{t+1}\})$$

We will exit the while loop if ~~strict~~ equality occurs.

∴ F is always decreasing.

∴ We will never repeat any iteration. ☺

∴ Converges to locally optimal solution

→ Probabilistic View of Clustering

Each cluster is a distribution.

Given 1) No. of clusters = k

2) Parametric form of distribution characterizing each cluster (eg - Gaussian)

eg - $f(x^i, \theta_k) \equiv$ density for cluster k

$\equiv N(x^i, \mu_k, \Sigma_k)$

$$= \frac{1}{(2\pi)^{d/2} |\Sigma_k|^{1/2}} e^{-\frac{1}{2}(x^i - \mu_k)^T \Sigma_k^{-1} (x^i - \mu_k)}$$

3) $D \equiv (x^1 \dots x^N)$

Mixture Distribution $\equiv P(x) = \sum_{k=1}^{k} \pi_k P(x|k)$

such that $\pi_k \geq 0$, $\sum \pi_k = 1$

$$= \sum_{k=1}^{k} \pi_k f(x^i, \theta_k)$$

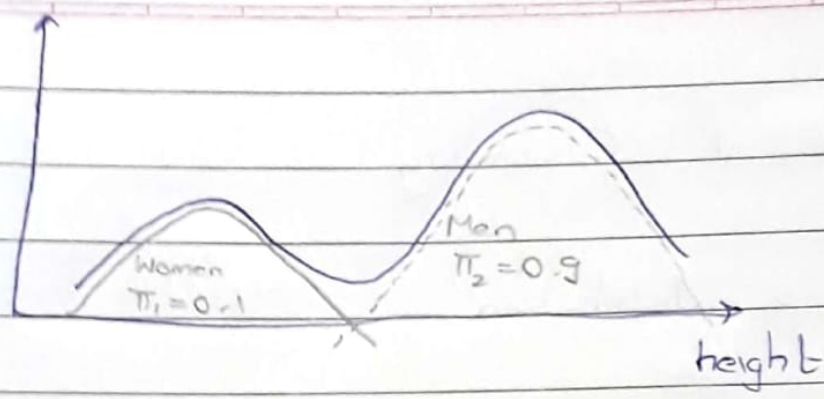
Values of π_i and θ_i for $i \in \{1 \dots k\}$ have to be discovered during clustering by MLE estimation.

→ Training objective :- Maximize $\sum_{j=1}^{N} \log P(x^j)$ over π_i, θ_i for $i \in \{1 \dots k\}$

P.T.O.

PAGE No.
DATE / /

eg



We ~~want~~ to ~~solve~~ find out π_1 & π_2 and θ_1 & θ_2