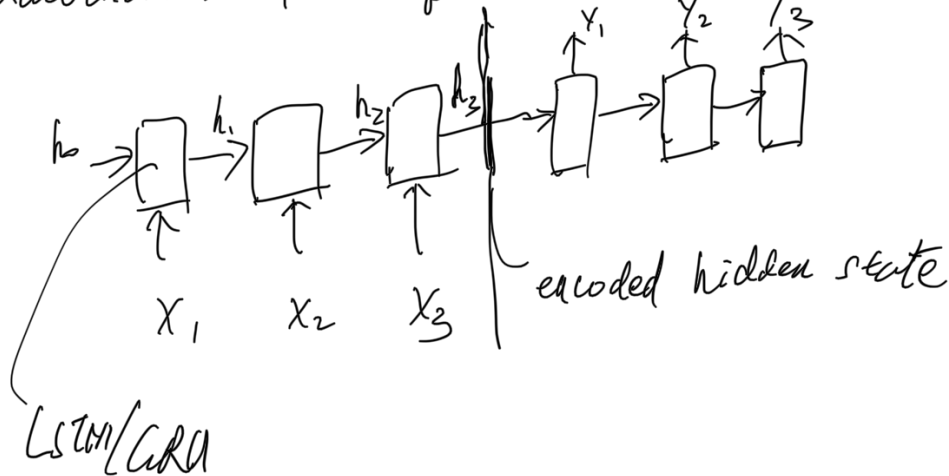


## Lecture 10

Autoencoder for Sequential Data



Dict: 10 words  $\Rightarrow$  represent all words with 10-D vectors

$$\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$$

Representation can not be very long — not efficient  
relationships between words { similarity / distance  
analogies.

Embedding

co-occurrence Matrix with Singular Value Decomposition (SVD)

$\downarrow$        $\downarrow$   
 We learn Deep Learning use A-Got  
 in r , , , 0 , 1 1 1

→ we

	0	1	2	1	0	1
learn	1	0	2	4	1	2
Deep	1	<del>2</del>	0	10	1	2
learning	0	2	4	1	0	6
else	1	0	1	1	0	0
A-lot	1	1	2	2	0	0

we learn deep learning

we use deep learning a lot

learning deep learning #

we learn a lot deep learning

$$X = USV^T$$

similar to PCA

Word2Vec :

- Continuous bag of words (CBOW)

- skip gram

prepare data : corpus , vocabulary , context length

CBOW :

- h  $\prod$

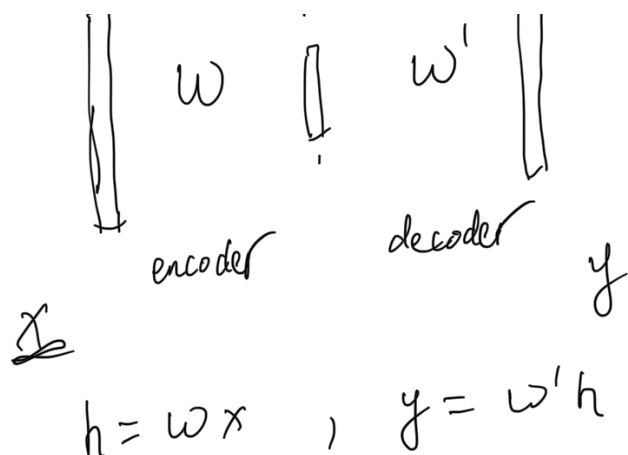


Diagram illustrating the calculation of the hidden state  $h$  as a weighted sum of input elements. A vertical vector of input elements is multiplied by a weight vector  $w$  to produce  $h$ .

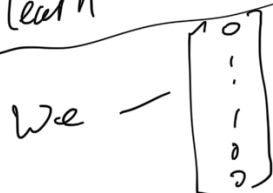
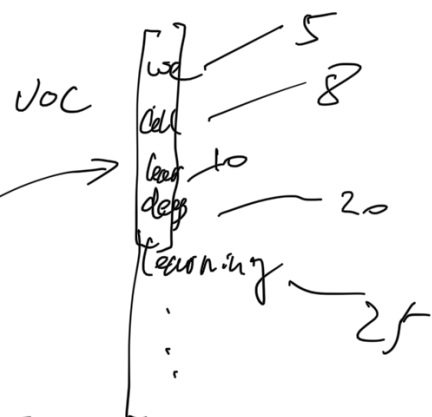
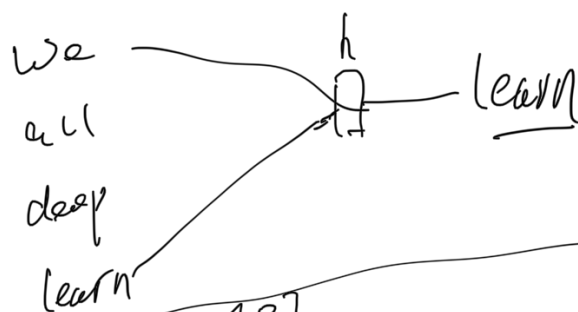
$$h = w(:, i)$$

Diagram illustrating the calculation of the hidden state  $h$  as a weighted sum of input elements. A vertical vector of input elements is multiplied by a weight vector  $w$  to produce  $h$ .

$$h = w(:, i)$$

word in  $Voc$ , index is  $i \Rightarrow x = \begin{bmatrix} 0 \\ \vdots \\ i \\ \vdots \\ 0 \end{bmatrix}$   $i$ -element  
 if word  $j$  in  $Voc$  is a neighbor of word  $i$ , related.

We all learn deep learning



$h$

$$\begin{bmatrix} w \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} \end{bmatrix} = \frac{w(:, 5)}{f}$$

we

$$w \begin{bmatrix} \end{bmatrix} \rightarrow \frac{w(:, 8)}{f}$$

all

$$w \begin{bmatrix} \end{bmatrix} \rightarrow \frac{w(:, 20)}{f}$$

deep

$$w \begin{bmatrix} \end{bmatrix} \rightarrow \frac{w(:, 25)}{f}$$

learning

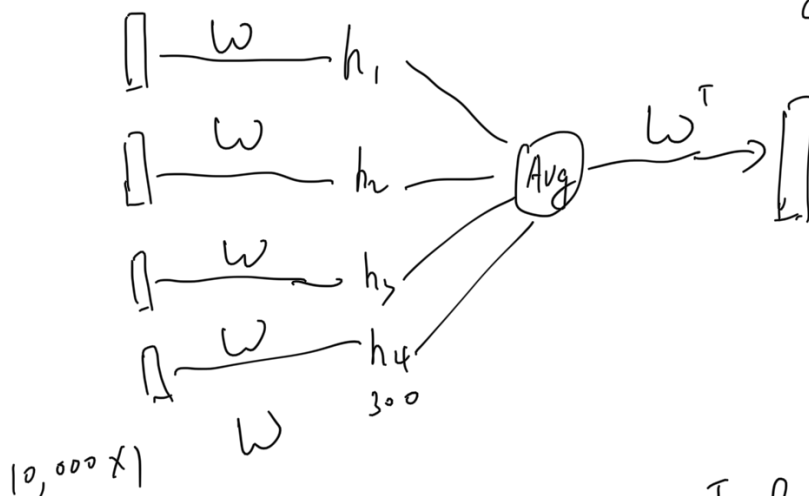
$$\underline{w}^T \underline{h} = \underline{y} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ 0 \\ 0 \end{bmatrix} \text{ 10-element}$$

softmax

$$\min (\hat{y} - y)^2$$

context words

center word



$$y = w^T h$$

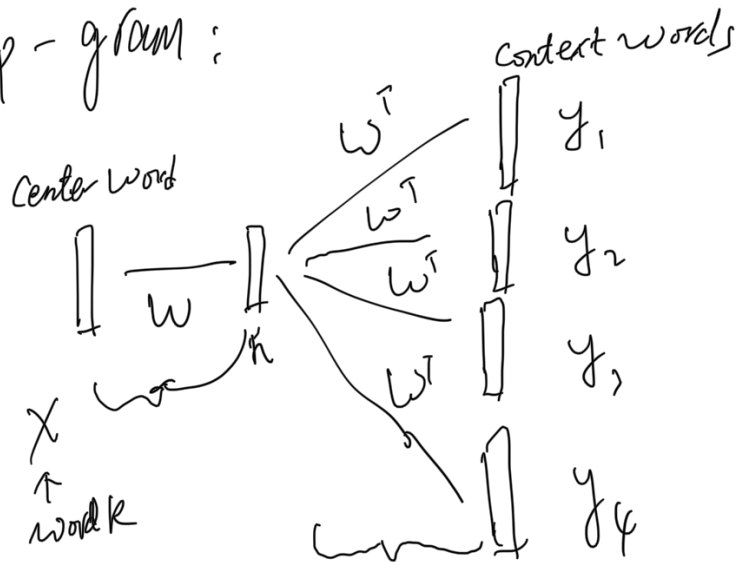
$$h = W X$$

$300 \times 1$        $10,000 \times 1$        $10,000 \times 300$        $300 \times 1$   
 $300 \times 10,000$        $10,000 \times 1$

After training,  
for each word

$$h = W X$$

skip-gram:



$$h = W X$$

$$\begin{bmatrix} w \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = W(:, k) \quad \text{X is vocab word } k$$

$$y = W^T h$$

$y$  — ground truth is one-hot

Diagram showing the calculation of  $y$  from  $h$  and  $W^T$ , resulting in a one-hot vector  $y$ .

$$w^T(j, :) h = y(j)$$

$$T = 10,000$$

maximize  
softmax

$$\frac{e^{w^T(j, :) h}}{\sum_{i=1}^I e^{w^T(i, :) h}}$$

$$\sum_j \frac{e^{w^T(j, :) h}}{\sum_{i=1}^I e^{w^T(i, :) h}}$$

$j \in$  words nearby  
voc word  $k$

we learn deep learning

