Notes About Word2Vec:

→ Word2Vec → - الـ ML والـ DL مبيفهمش انا نديلهم الـ text
فلازم نعبر عن الـ Text بشكل Numerical.

→ tf-idf & Counts doesn't perserve any relationship between the words & this is where the word embeddings come in.

← كده الـ tf-idf بيقولك الكلمة اتكررت كام مرة في كام document لكن مش بيقولك ازاي مرتبطة بالكلمات التانية.

→ PPMI perserve the relationship between words depending on sparse matrix, but we want to have a dense matrix to represent the relations, more over in word2vec we use NN, while PPMI uses classical methods such as counting.

كدا ✓ make sense. عشان الـ ~~PPMI~~ word2vec بيقدر يفهم الـ context اللي انت شغال فيه على عكس الـ PPMI اللي مش بيفهم الـ context اوي كل قد ما بيفهم ان فيه كلمتين مترابطين او بيجيوا مع بعض كتير.

- الفكرة الأساسية من الـ word2vec ان اعبر عن الـ word في شكل vector والـ vector ده يكون جواه المعنى بتاع الكلمة دي. ومش بس المعنى، 'لا' المعنى بتاعها في كل سياق.

وعندنا نوعين من الـ word2vec →
- Continuous Bag of words
- Skip Gram

we use only 2 layer NN

famous example
"King - men + women = Queen"

CBOW Example:

Corpus Vocabulary are {"word", "vectors", "are", "widely", "used"}

∴ $|V| = 5$ # of unique words.

⇒ Assume that the given context is 1 ∴ $C = 1$ & we Predict the next word.

if the training sentence was "word vectors are widely used" so if word is "word" then the target will be "vectors".

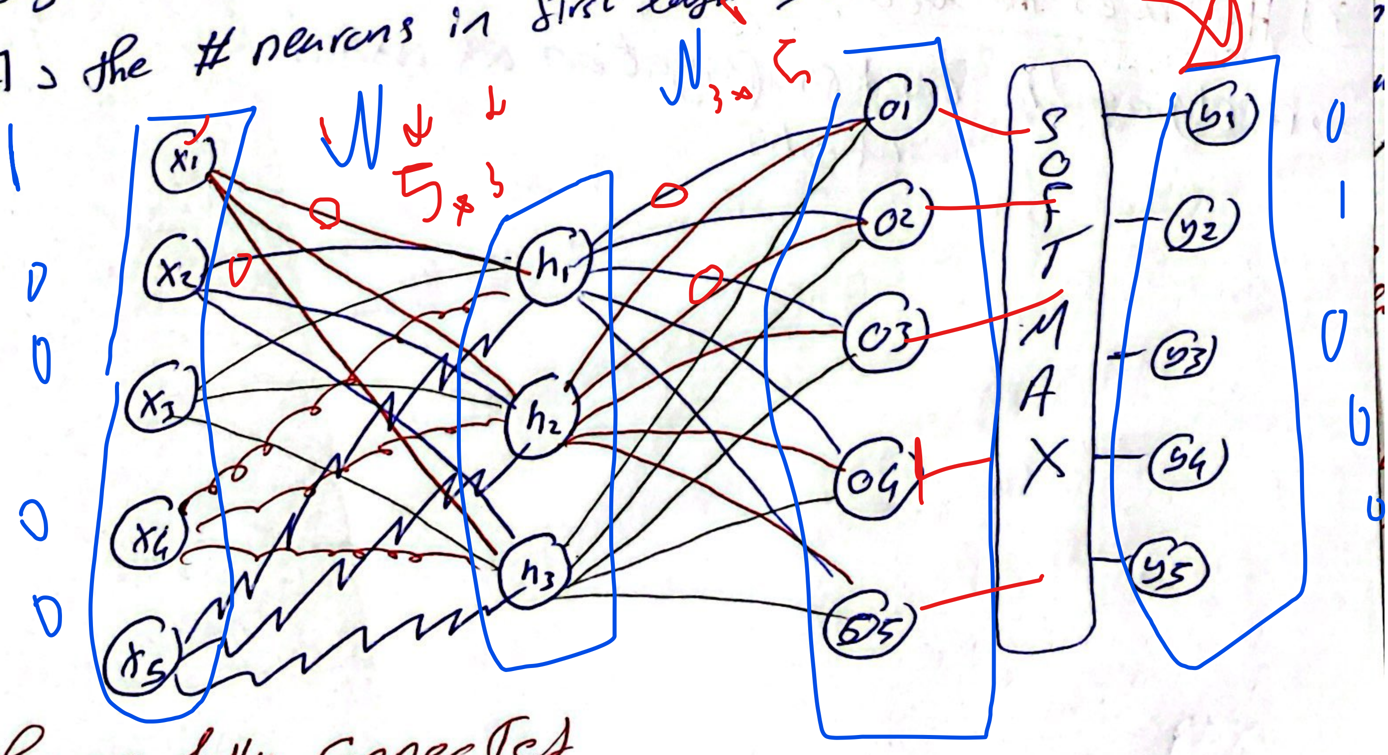
// Now lets get the word encoding useing one hot vector.

$$\underset{\text{word}}{\begin{bmatrix}1\\0\\0\\0\\0\end{bmatrix}} \quad \underset{\text{vector}}{\begin{bmatrix}0\\1\\0\\0\\0\end{bmatrix}} \quad \underset{\text{are}}{\begin{bmatrix}0\\0\\1\\0\\0\end{bmatrix}} \quad \underset{\text{widely}}{\begin{bmatrix}0\\0\\0\\1\\0\end{bmatrix}} \quad \underset{\text{used}}{\begin{bmatrix}0\\0\\0\\0\\1\end{bmatrix}}$$

// Now lets build the three (two) layered NN:

[1] we need to determine # of neurons in the hidden layer, so we can assume for this Problem that we 3

[2] ⇒ the # neurons in first layer 5 - second = 3, output = 5



they are fully connected

Ahmed is playing football

[3] Now we need to define the Matricies

→ // In this lecture we follow different Convention where the dimension of the matrix is reversed

→ So the Input hidden layer matrix will be $W_{V\times N}$ instead of $W_{N\times V}$, where w_{ji} = weight between x_j & h_i

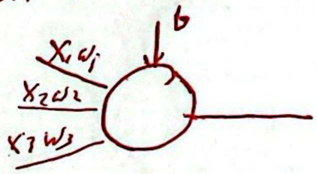
$$
\begin{array}{c|ccc}
 & h_1 & h_2 & h_3 \\
\hline
x_1 & w_{11} & w_{12} & w_{13} \\
x_2 & w_{21} & w_{22} & w_{23} \\
x_3 & w_{31} & w_{32} & w_{33} \\
x_4 & w_{41} & w_{42} & w_{43} \\
x_5 & w_{51} & w_{52} & w_{53}
\end{array}
\quad V
$$

// follow the following scheme

$$
\begin{array}{c|ccccc}
 & O_1 & O_2 & O_3 & O_4 & O_5 \\
\hline
h_1 & w'_{11} & w'_{12} & w'_{13} & w'_{14} & w'_{15} \\
h_2 & w'_{21} & w'_{22} & w'_{23} & w'_{24} & w'_{25} \\
h_3 & w'_{31} & w'_{32} & w'_{33} & w'_{34} & w'_{35}
\end{array}
$$

→ Same for the Hidden output layer matrix it will be $W_{N\times V}$ instead of $W_{V\times N}$, where w'_{ij} represent the weight between h_i & O_j

[4] Now we need to evaluate each neuron in the hidden layer.



// Remember that

→ we follow the basic Convention of NN, but here we neither add bias, nor use non-linear activation function

→ So it is just a weighted sum

$$\therefore h_i = \sum_j w_{ji} * x_j$$

i.e. $h_1 = w_{11}x_1 + w_{21}x_2 + w_{31}x_3 + w_{41}x_4 + w_{51}x_5$

→ the Column in the first matrix h_1

& the same convention for h_2 & h_3

[2]

5) Now we need to evaluate the neurons of output

→ Following the same convention, we get this equation

$$O_j = \sum_i w_{ij} * h_j$$

follow the same convention for all other Neurons O_{1-5}

ie:

$O_1 = w_{11} h_1 + w_{21} h_2 + w_{31} h_3$

$O_2 = w_{12} h_1 + w_{22} h_2 + w_{32} h_3$

Culmn of O_i in the matrix $[\,]_{V}$

---

6) Now we need to insert all O_i to softmax layer just to get the target word.

$$y_i = \text{softmax}(O_j),\ j \in [1-5]$$

Note: we don't care much about the output because we just need to extract the word embeddings which are represented in the weights of the Matricies.

---

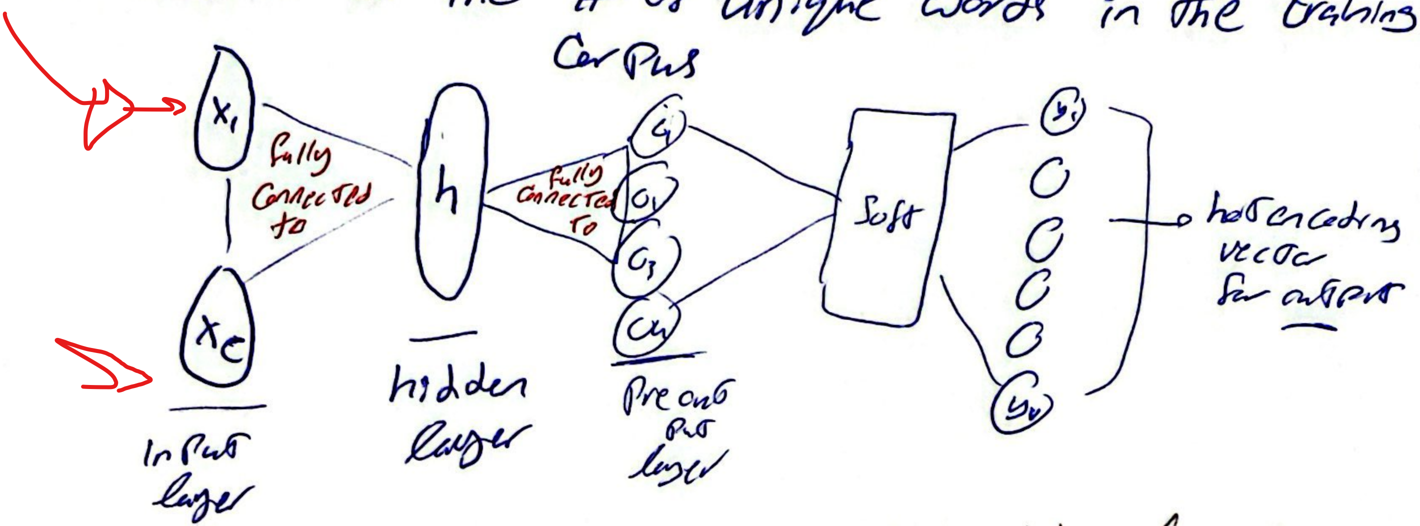
7) How does the Network learn?

→ Applying the backpropagation as usual to adjust weight

4) For CBOW model, if we have number of words in vocab V & the dimension of the embedding is $\underline{d}$ (CBOW Problem in sheet)

i) draw the Network.

// Remember that the Input layer is CV, where C is the Context words & V is the size of each vector which is similar to the # of unique words in the training corpus



ii) what are the # of neurons in the hidden layer?

→ It is given in the problem → d = # of embeddings = # of neurons in hidden layer.

iii) what are the dimensions for weight matrix between hidden layer & Input

→ we will have C matricies ← not sure ↳ I think not

→ each of them will be of size $W_{V \times d}$

Note → here we following different convention rather than the normal convention in neural networks, as usually we would make it $W_{d \times V}$.

iv) what are the dimension for weight matrix between hidden & output

→ It would be $M_{d \times V}$

v) what are dimension of input/output vectors?

$\boxed{V \times 1}$