Continuous Bag of words model

- Consider the task: Given (n-1) previously observed words, predict the n^{th} word.

Example: He sat on a chair : Think of this as a Classification problem.
(sat on a = x, chair = y)

Y or the output is a one-hot encoded vector of size equal to the size of the vocabulary.

Training Data: From the given corpus, construct a n window training data.

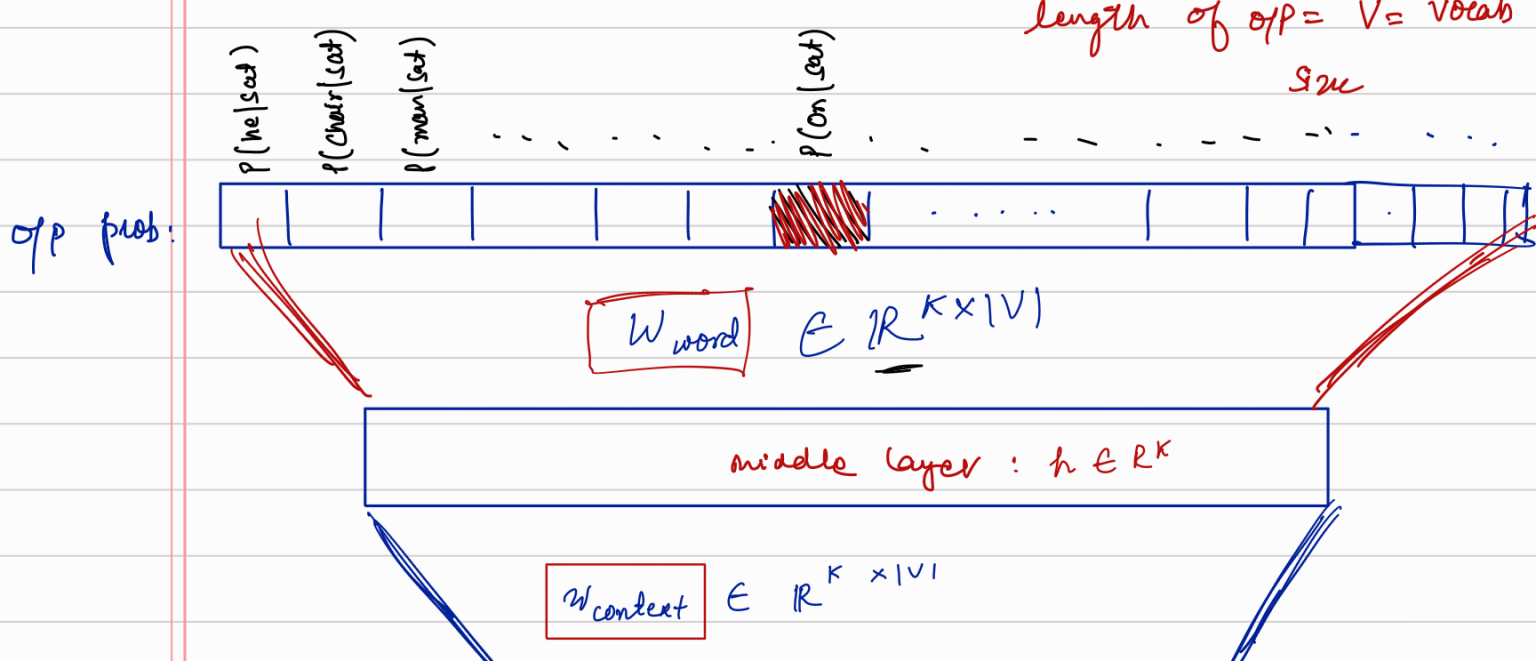
Corpus: Hello, my name is Sahil. I am 22 years old male from India

Given: my, name, is : predict SAHIK.
Window 2

Case when n = 2

Input: one word: 'SAT' : | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

← OHE Rep of input word



SAT : | 0 | 0 | 1 | 0 | ...... | 0 | 0 | OHE.

Also V dimensional $x \in \mathbb{R}^{|V|}$ { one-hot encoded representation }

Wcontext : Dimension = $(K \times V)$

$$\begin{bmatrix} K \times V \end{bmatrix} * [V \times 1] = [K \times 1]$$

Wcontext   Input

- Let's explore the $W_{context} \times$ 'X'

$$\begin{bmatrix} \uparrow & \uparrow & & \uparrow & \uparrow \\ w_1 & w_2 & \cdots & w_K & w_V \\ \downarrow & \downarrow & & \downarrow & \downarrow \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} \vec{w}_K \end{bmatrix}$$

K^{th} element = 1

{ Each $\vec{w}_i$ = K dimensional vector
There are V such vectors. }

main takeaway :→ Since the input is one-hot encoded, the product of the two will be a the K^{th} column always.

Input one-hot encoding acts as a switch where depending on which element = 1, a column from Wcontext will be selected.

There is 1:1 mapping between number of words in Vocab & the columns of the Wcontext matrix.

- How do we obtain the required output i.e. P(on | sat)

we use the softmax:

$$P(on \mid sat) = e^{W_{word} \times h\,[i]}$$

↳ ith Entry

] get the output and then

$$\frac{e^{W_{word}\, h[i]}}{\sum_{i=1}^{V} e^{W_{word}\, h[i]}}$$

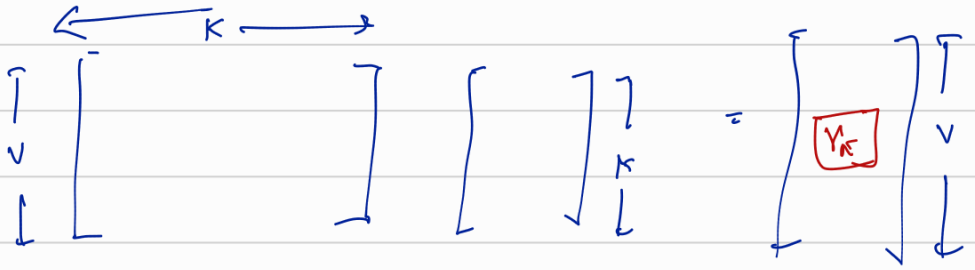
apply a softmax transformation

hidden layer "h" $\in \mathbb{R}^K$

opp layer = $W_{word} * h \longrightarrow$ Softmax Transformation

opp = W_{word} $\times$ h ~ a column of W context depending on what input is given.

$(V \times K)$ $(K \times 1)$



Each Entry in the o/p vector of length V

V_K: K^{th} Entry in the opp vector = dot product of K^{th} row of W_{word} and the j^{th} column of $W_{context}$. 'j's value is determined by which element of the OHE vector is equal to '1'.

this is because hidden layer is created based on OHE vector

Thus $P(word = i \mid sat)$ thus depends on the i^{th} Row of W_{word}.
Thus W_{word}'s ith Row also corresponds to a particular word.

So, the model learns two different representations. One when word appears as context word & one when it appears as a target word.

Further Exploration

Let's denote the context word (sat) by C and the correct output word (on) by the index 'w'

loss function for the multiclass problem formulation: Cross-entropy

$$L(\theta) = -\log(\hat{y}_w) = -\log P(w|C)$$

It is equivalent to minimizing just the score of 'w' or $p(on \mid sat)$

Cross entropy = $-\sum_{i}^{n} y_i \log(p_i)$ $y_i = 1$ only for 'on' so it

reduces to just $-\log(\hat{y}_w)$

$h = W_{context} \times x_c = u_c$] Basically the column of $W_{context}$, let's say W_j, based on what 'j' is in the one hot rep of the word.

vectorial representation of the word 'sat'.

$$\hat{y}_w = \frac{\exp(u_c \cdot V_w)}{\sum_{w' \in V} \exp(u_c \cdot V_{w'})} \quad (= K)$$

] same as mentioned above. $u_c \cdot V_w$ = product of two columns.

$$L(\theta) = -\log(\hat{y}_w) = -[\log(\exp(u_c \cdot V_w) - \log(K)]$$

$$L(\theta) = -u_c \cdot V_w + \log\left(\sum_{w' \in V} \exp(u_c \cdot V_{w'})\right)$$

$$\frac{\partial L}{\partial V_w} = -u_c + \left[\frac{1}{K} \times \exp(u_c \cdot V_{w'}) \times u_c\right]$$

↳ softmax again

V_w is parameter that we intend to learn.

$$\frac{\partial h}{\partial V_w} = -u_c \cdot (1 - \hat{y}_w)$$

update Rule

$$V_w^{new} = V_w^{old} - \eta \times \frac{\partial h}{\partial V_w}$$

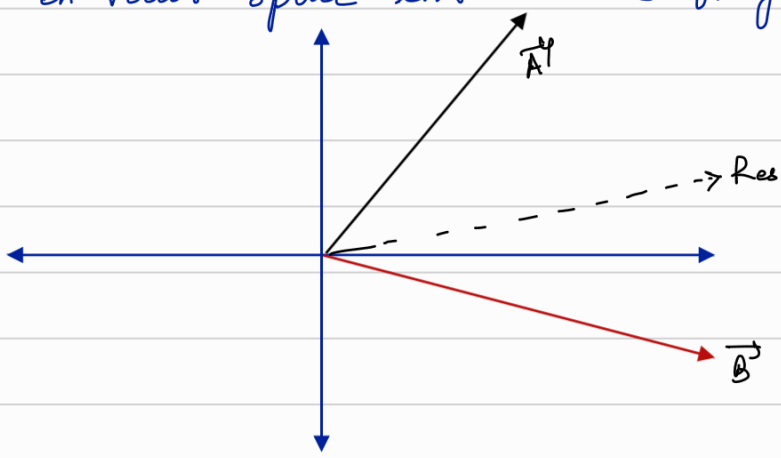
$$V_w^{new} = V_w + \eta\, u_c (1 - \hat{y}_w)$$

Case 1: $\hat{y}_w = 1$] This is the ideal case, meaning that we have learnt a good representation of V_w

Case 2: $\hat{y}_w = 0$] This case, the new representation of V_w is

$V_w^{new} = V_w + \eta\, h_c$

In vector space this means shifting V_w closer to h_c.

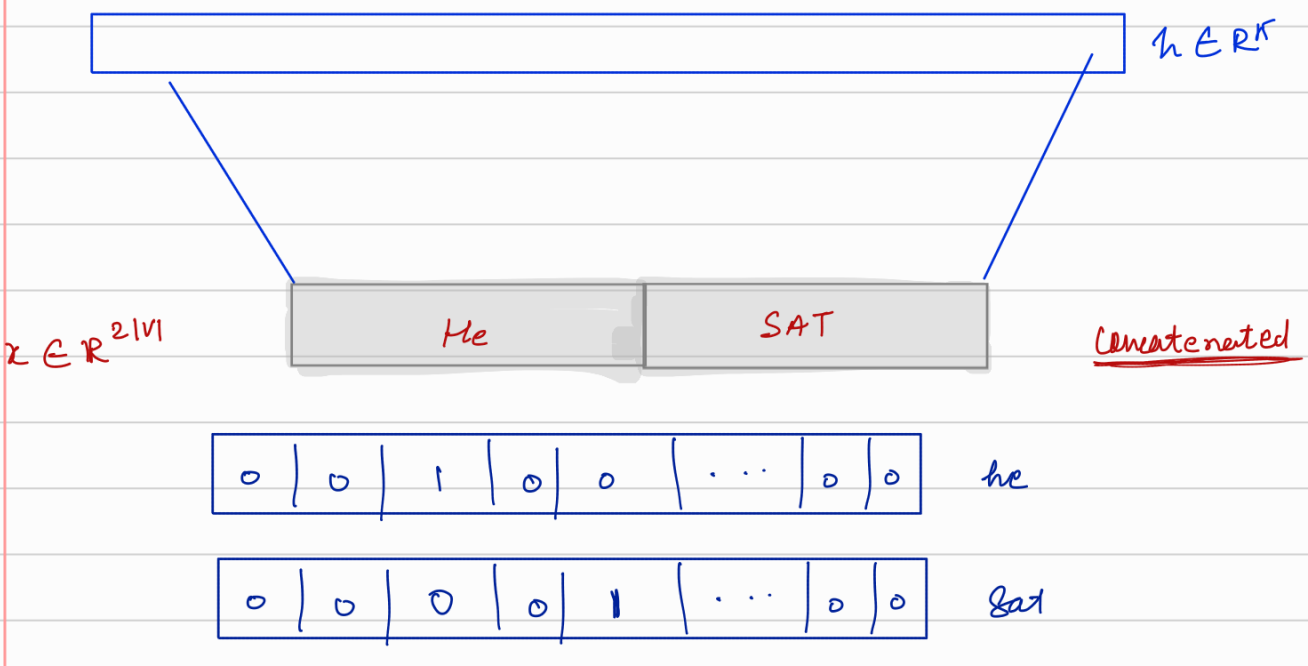


$\vec{A} + \vec{B}$ = A vector that is closer to both $\vec{A}$ & $\vec{B}$

Essentially: The algorithm is just trying to make word representation closer to the context representation, thereby ensuring that the cosine similarity between the two vectors is maximized.

- Lets expand on this a little bit: What happens if we change the window to '3'. Meaning, Given 2 words, predict the third word.

$P(on \mid \{the, sat\})$



$h \in \mathbb{R}^K$ is simply the sum of all the columns that are '1' in the one-hot encoded realm.

For example if he and sat are input, lets say col 5 & col 7. h = Sum (col 5 & col 7) for the $W_{context}$.

During the back propagation, All columns of W_{word} and 2 columns of the $W_{context}$ will be updated.

Issues with this case

Here, the pure issue is with the softmax function.

$op = \mathbb{R}^V = (1 \times K) \times [K \times V]$ matrix.

passing to the softmax.

Each entry j, lets call it $p_j = \dfrac{e^{(h \times W_{word}[:,j])}}{\sum_{k=1}^{V} \exp\left(h \times W_{word}[:,k]\right)}$

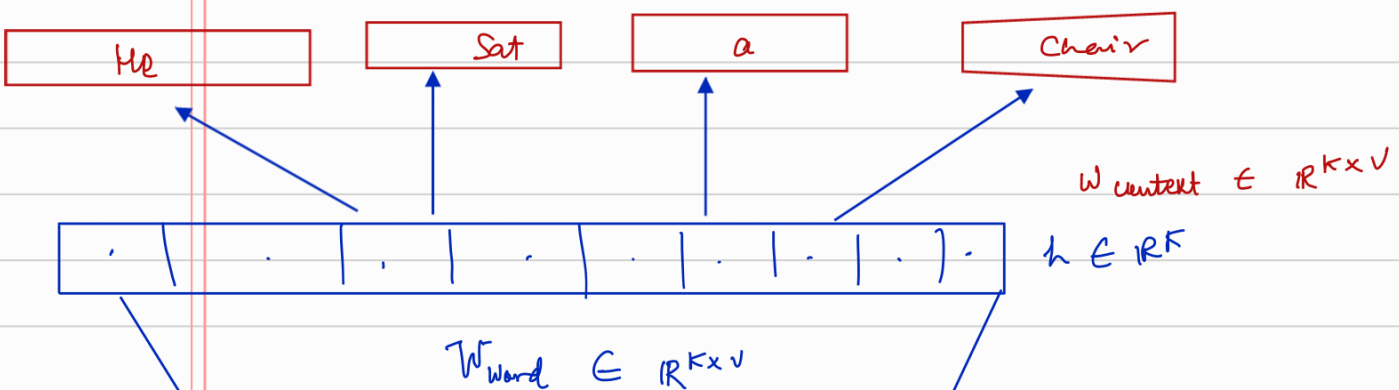
Can be of the order of millions.

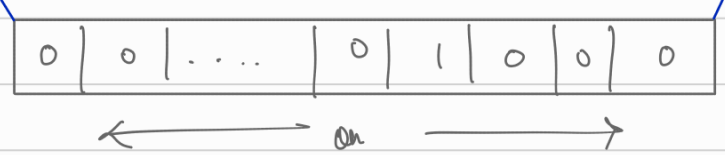
Computationally complex

Skip-Gram model

This model is kinda reverse of the previous CBOW model.

objective: Given a word, predict a set of context word.





$x \in \mathbb{R}^{|V|}$

← Set of input word → obviously it will be a V-dimensional vector which is one-hot encoded.

- Loss function: Set of 4 categorical cross entropies
- Also, we have flipped W_{word} and $W_{context}$ in this case.

$$\text{loss function} = \mathcal{L}(\theta) = -\sum_{i=1}^{d-1} \log(\hat{y}_{w_i})$$ → probability

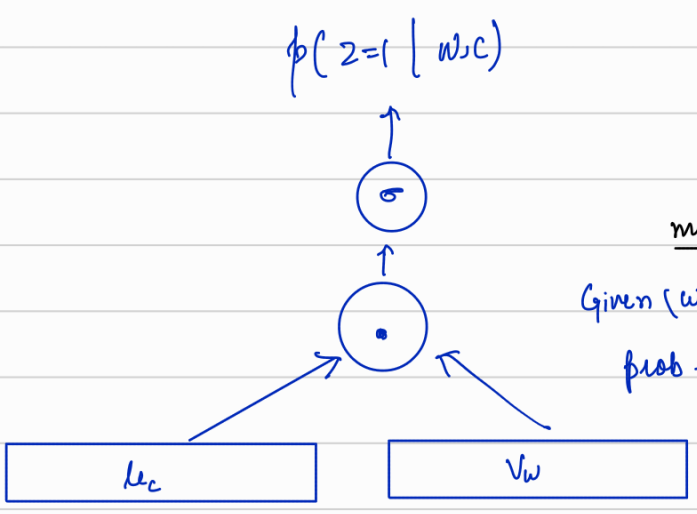
d = 5 in our case, which is the window.

Loss function = Sum of 4 categorical cross entropies

Some problems: Same issue as bag of words. Instead of one softmax, now we have four softmax which adds a lot to the computational issue.

Solution 1: use negative sampling:
Sol 2: Contrastive estimation
Sol 3: hierarchical softmax.

→ Negative Sampling: Let D be a set of corpus D = Set (w, c) pairs
D' = (w, z) → Synthetically created pairs that don't belong together.



V_w = Rep. of word
l_c = Rep. of context word

model of the problem

Given (w, c), I want to maximize the prob. $p(z=1 \mid w, c)$

$$p(z=1 \mid w, c) = \sigma(l_c^T V_w)$$

Consider $(w, c) \in D$, I want to maximize the $p(z=1 \mid w, c)$

$$\text{Loss}(u, v) = - \prod_{(w,c) \in D} p(z=1 \mid w, c)$$

$$\boxed{\log(\text{Loss}) = - \sum_{(w,c) \in D} p(z=1 \mid w, c)}$$

Simultaneously $\forall\ w, r \in D'$, I want to max. $p(z=0 \mid (w, r))$

$$\text{objective} = \max \prod_{(w,r) \in D'} p(z=0 \mid (w, r))$$

objective: For every pair in 'D' max $\prod p(z=1 \mid (w,c))$ AND for every pair in D', max $\prod p(z=0 \mid (w, r))$

$$\text{obj} = \max \left[\underbrace{\prod_{(w,c) \in D} p(z=1 \mid (w,c))}_{A} \times \underbrace{\prod_{(w,r) \in D'} p(z=0 \mid (w,r))}_{B} \right]$$

prob is modeled as a Sigmoid function

$$p(z=1 \mid (w,c)) = \sigma(u_c^T v_w)$$

u_c: Vector rep. of context

$$\log(\text{objective}) = \sum_{(w,c) \in D} \log(\sigma(u_c^T v_w)) + \sum_{(w,r) \in D'} \log(1 - \sigma(u_r^T v_w))$$

v_w: vector rep of word

u_r: vector rep of randomly selected negative word

$$1 - \sigma(u_r^T v_w) = 1 - \frac{1}{1 + e^{-\sigma \times u_r^T v_w}}$$

$$= \frac{1 + e^{-\sigma \times u_r^T v_w} - 1}{1 + e^{-\sigma u_r^T v_w}} = \frac{1}{1 + e^{\sigma u_r^T v_w}}$$

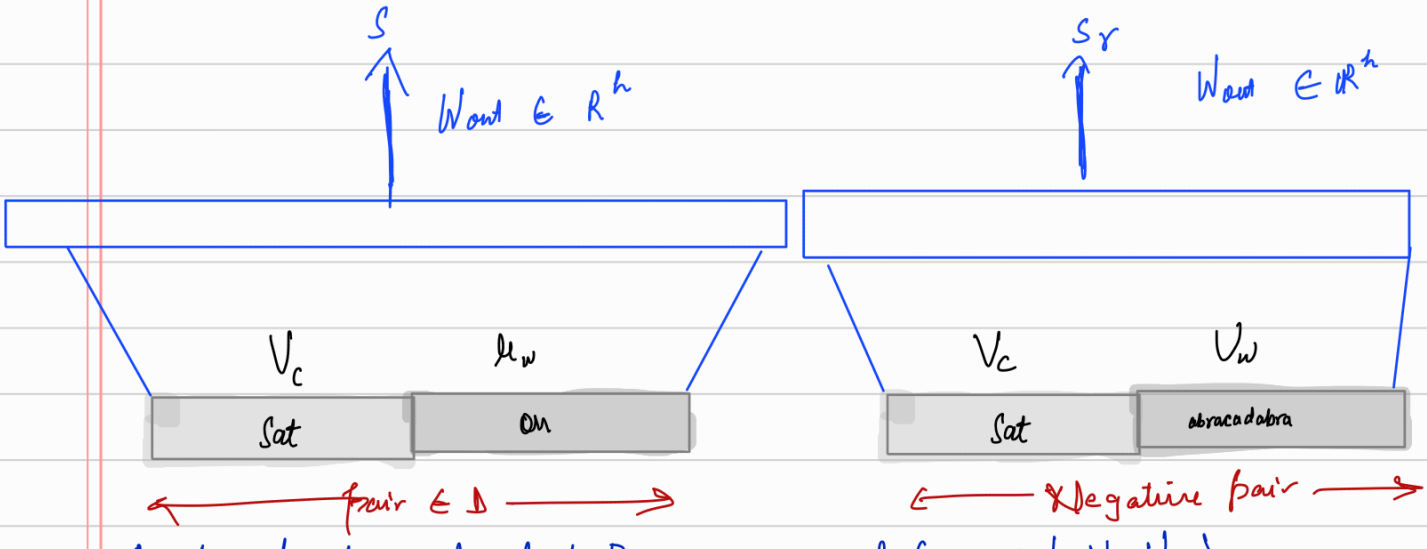
$$\log(\text{objective}) = \sum_{w,c \in D} \log\left(\sigma(u_c^T v_w)\right) + \sum_{w,r \in D'} \log\left(\sigma(-u_r^T v_w)\right)$$

($u_c^T v_w$: dot product; $u_r^T v_w$: dot product)

Key Takeaway :- In addition to bringing close words together (part A) of above Eqn, we are also moving dissimilar words away from each other (part B) of Eqn.

Contrastive Estimation

Instead of formulating the problem as a probability problem like above, we can formulate it as following:



Instead of calculating the $p(z=1 \mid V_c, V_w)$ we can train it as a regression problem where one has to calculate the output score 'S'.

We would like for S to be much greater than S_r. Not only that, I want S to be greater than S_r by a margin

$S - S_r$ should be at least 'm'] m will be controlled by me.

Thus, we are interested in $S - (S_r + m)$

loss = if $S > S_r + m$; $L = 0$

$S < S_r + m$; calculate loss and adjust your embeddings.

$$\text{loss} = \max(0, \; S - (S_r + m))$$

→ no softmax use here.