Stanford CS224 notes :-

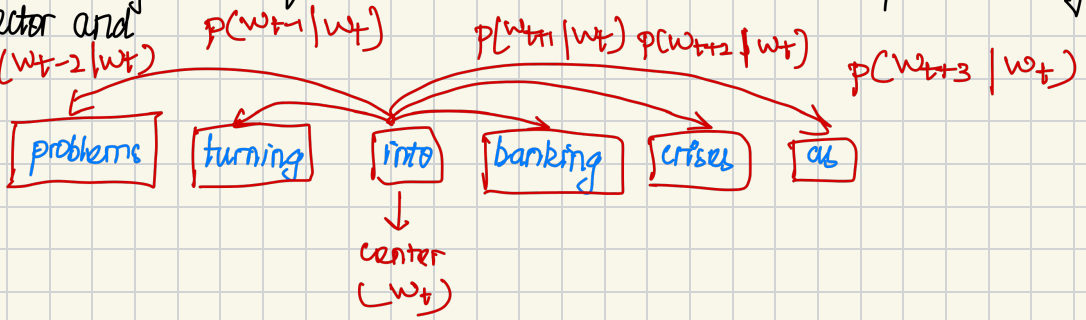
words are represented as a set of numbers by the computers but these numbers are for telling the computer to turn on which pixel in the display to display the particular word.

Word vectors :-

basically here we are turning all the words into multi dimensional vectors while each dimension recording a specific context in which the words occur in same context.

Word2Vec :-

Have a long list of words. Each word should be represented by a vector and



Likelihood function

$t \rightarrow$ position of a particular word which is under consideration (or) comparison.

$m \rightarrow$ this is the context window size basically we consider m words before and after "t"

$p \rightarrow$ probability of the word occuring next to "t".

$$L(\theta) = \prod_{t=1}^{T} \prod_{-m \le j \le m,\ j \neq 0} p(w_{t+j} | w_t, \theta)$$

Because we are using π for likelihood we have to use

Once we take log of likelihood then it becomes like a sum which is computationally easier than product.

objective function =

$$j(\theta) = -\frac{1}{T}\log L(\theta) = \frac{-1}{T}\sum_{t=1}^{T}\sum_{-m\le j\le m}\log P(w_{t+j} \mid w_t ; \theta)$$

we are using $\frac{1}{T}$ because to control the size (or) value and making it small.

how to calculate $P(w_{t+j} \mid w_t ; \theta)$

we will use 2 vectors for a word "w"

V_w when w is center word.
U_w when w is context word.

$$P(o|c) = \frac{\exp(u_o^T v_o)}{\sum_{w\in V}\exp(u_w^T v_c)}$$

$$u_o^T v_o \Rightarrow u^T v = u \cdot v = \sum_{i=1}^{n} u_i v_i$$

larger the dot product → larger the probability.

Softmax function $(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{n}\exp(x_j)} = p_i$

$R^n \to (0,1)^n$

Dissection of Softmax.

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

exp because we need positive values for probability calculation.

$u_w^T v_c$ → dot product for vector values of word "w" as a context word with "c" as the center word.

Bigger the dot product ↓ higher will be the similarity.

probability of context word occuring near to the center word.

overall we will get the probability distribution of the context word with the center word according to all the words in the vocabulary "V"

Derivative formulas :

$$\frac{\partial}{\partial x}(a \cdot x) = a \rightarrow \text{is constant.}$$

So if $\frac{\partial}{\partial v_c}(-u_o^T v_c)$ then it is $-u_o$ which is a constant.

log and exp.

$$\log(\exp(u_o^T v_c)) \Rightarrow u_o^T v_c$$

because log and exp are inversely correlated.

So now we will apply derivative on softmax to simplify.

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

loss function is given by $L = -\log P(o|c)$

we are using "$-$" because we want a positive number of how much we are wrong.

we are using "log" because we want the probability number to be in a bigger range than just $[0, 1]$ because then we will be able to tell the model how much it is wrong very easily. with a very small range of $[0, 1]$ we will not be able to do it.

$P(o|c)$ → probability of context word "o" to appear next to center word "c".

$$L = -\log\left(\frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}\right)$$

As per $\log\left(\frac{a}{b}\right) = \log(a) - \log(b)$ we can say

$$L = -\left[\log(\exp(u_o^T v_c)) - \log\left(\sum_{w \in V} \exp(u_w^T v_c)\right)\right]$$

what is a log?

log tells us how much should I raise the power of e to get a number x

$\log(x)$ gives this num → $e^? = x$

Now if we are saying $\log(a+b+c) \neq \log(a) + \log(b) + \log(c)$

because in $\log(a+b+c)$ we are asking what is the power of e to get a number "a+b+c" but when we say $\log(a) + \log(b) + \log(c)$ then we are asking for each number "a" "b" & "c".

So $\log(a+b+c) \neq \log(a) + \log(b) + \log(c)$

Now considering this we can say.

We cannot apply logarithm inside a summation. The sum has to be done first and then the log has to be performed.

$$L = -\left[\log(\exp(u_o^T v_c)) - \log\left(\sum_{w \in V} \exp(u_w^T v_c)\right)\right]$$

now keeping that in mind

$$L = -u_o^T v_c + \log\left[\sum_{w \in V} \exp(u_w^T v_c)\right]$$

Now we have the loss function and we have to understand how we much of a change in v_c is affecting the loss function "L"?

So we take derivative with respect to v_c $\frac{\partial}{\partial v_c}$

$$\frac{\partial L}{\partial v_c} = \frac{\partial}{\partial v_c}(-u_o^T v_c) + \frac{\partial}{\partial v_c} \log\left[\sum_{w \in V} \exp(u_w^T v_c)\right]$$

$$\frac{\partial}{\partial x}(a.x) = a \rightarrow \text{this is a rule}$$

which says that in an expression $(a.x)$ we just have to adjust x "a" times to make a difference to the value of $a.x$

So applying this rule we can say that

$$\frac{\partial}{\partial v_c}(-u_o^T v_c) = -u_o$$ So now we have.

$$\frac{\partial L}{\partial v_c} = -u_o + \frac{\partial}{\partial v_c} \log\left[\sum_{w \in V} \exp(u_w^T v_c)\right]$$

$\frac{\partial L}{\partial v_c}$ → it's the rate of change of loss function L with change in embeddings of center word.

$\frac{d}{dx}[5x+3]$ Here we can say $\frac{d}{dx}[f(g(x))]$ ↓ $f'(g) \cdot g'(x)$

So $f'(g) = [5x+3]^4$

$g(x) = 5x+3$

So lets calculate $f'(g)$

$$= 4[5x+3]^3 \times [5]$$

$$= 20[5x+3]^3$$

So the derivative of constant is 0 and for a variable is 1

So here $\frac{d}{dx}[f(g(x))] = f'(g) \cdot g'(x)$

is called the chain rule probably the most important rule when it comes to model training.

$$\frac{\partial L}{\partial v_c} = -u_o + \frac{\partial}{\partial v_c} \log\left[\sum_{w \in V} \exp(u_w^T v_c)\right]$$

now lets continue solving the 2nd term $\rightarrow \frac{\partial}{\partial v_c} \log\left[\sum_{w \in V} \exp(u_w^T v_c)\right]$

now as per chain rule we can say

$$\frac{\partial}{\partial v_c} f(z(v_c))$$

$f'(z) \cdot z'(v_c)$ now lets break down and work on $f'(z)$

$$f'(z) = \frac{d}{dz}(\log(z)) = \frac{1}{z}$$

$$z'(v_c) = \frac{d}{dv_c}\left[\sum_{w \in V} \exp(u_w^T v_c)\right]$$

now lets dive into $\frac{d}{dv_c}\left[\sum_{w \in V} \exp(u_w^T v_c)\right]$

$g(v_c) \rightarrow$ inner function is $u_w^T v_c$

$f(g(v_c)) \rightarrow$ outer function is $\exp(v_w^T v_c)$

Now lets work out $g'(v_c)$

$$\frac{d}{dv_c}(u_w^T v_c) \quad \text{w.k.t} \quad \frac{\partial}{\partial x}(a \cdot x) = a.$$

using that we say $\frac{\partial}{\partial v_c}(u_w^T v_c) = u_w$

Now lets go to outer function.

$$\frac{\partial}{\partial v_c}\left[\exp(u_w^T v_c)\right] \quad \text{w.k.t} \quad \frac{\partial}{\partial x}\left[\exp(x)\right] = \exp(x)$$

so using that we say $\frac{\partial}{\partial v_c}\left[\exp(u_w^T v_c)\right] = \exp[u_w^T v_c]$

combining both results we can say $\sum_{w \in V} \exp[u_w^T v_c] \cdot u_w$

Now lets take the $\frac{1}{Z}$ also. so that whole expression comes to.

$$\frac{\partial L}{\partial v_c} = \frac{1}{Z} \cdot \sum_{w \in V} \exp(u_w^T v_c) \cdot u_w$$

$$= \frac{1}{\sum_{w \in V} \exp(u_w^T v_c)} \cdot \sum_{w \in V} \exp(u_w^T v_c) \cdot u_w$$

$$\frac{\partial L}{\partial v_c} = \sum_{w \in V} \frac{\exp(u_w^T v_c) \cdot u_w}{\sum_{w' \in V} \exp(u_{w'}^T v_c)}$$

taking this part seperate we can say that

$\frac{\exp(u_w^T v_c)}{\sum_{w' \in V} \exp(u_{w'}^T v_c)}$ is nothing but $p(w|c)$ probability of w occuring next to "c" word.

$$\frac{\partial L}{\partial v_c} = \sum_{w \in V} p(w|c) \cdot u_w = u_o$$

which essentially means the change in loss function w.r.t v_c can be achieved by doing a sum of all the multiplications between the probability of each word in the vocabulary happening next to center word "c"

But the problem with word2Vec embedding is because of the term $\sum_{w \in V} u_w^T v_c$ now because of the enormous data that we have this will take forever. Thats why we need a new algorithm which will help us to complete the training with less compute.

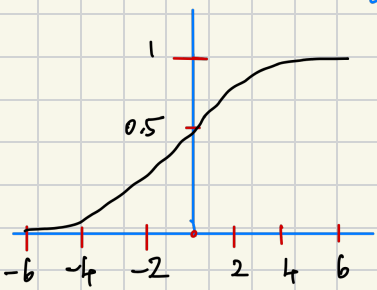
So we introduce something called stochastic gradient Descend (SGD)

Skip-Gram with negative sampling.

$$J_{neg\text{-}sample}(u_o, v_c, U) = -\log\sigma(u_o^T v_c) - \sum_{k \in (K \text{ samples})} \log\sigma(-u_k^T v_c)$$

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

sigmoid rather than softmax.



Sigmoid function always reduces the number passed to it within the range [0, 1] kind of like a probability.

lets say we get a output from a neuron which is a large number. and. if we keep going like that we will not have a easy classification function. For that we have to have curves and bends.

That is the reason why we use sigmoid, RELU, tanh kind of functions.