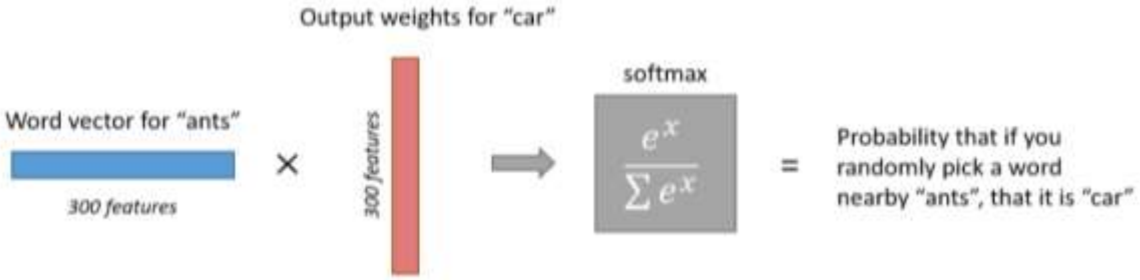
Assignment 2



Notations and Notes -

- Both the input vector x and the output y are one-hot encoded
- v_w and v'_w are two representations of the input word w
- v_w comes from the rows of W (Here V)
- v'_w comes from the columns of W' (Here U)
- v_w is usually called the input vector (Here v_w)
- v'_w is usually called the output vector (Here u_w)

A good doc to understand the inner workings - http://www.1-4-5.net/~dmm/ml/how_does_word2vec_work.pdf

Objectives –

Understanding Word2Vec (Skip Gram) Model

Understanding the Architecture of the model

Understanding the loss, error, various gradients, input layer, output layer, input vector and output vector, probability of context vector given centre vector.

And

Implementing Sigmoid Function

Implementing Softmax and Negative Sampling loss and their gradients

Implementing the loss and gradients for Skip-gram model

Implementing SGD Optimizer for back propagation

Lets start -

(a)

This problem can be simply solved looking at the 1 hot encoded nature of output word(context vector in this case).

I have elaborated the same below -

a) As described in the doc, y is a one-hot vector with a 1 for the true outside word o, that means y_i is 1 if and only if $i == o$. so the proof could be below:

$$-\sum_{w \in Vocab} y_w \log(\hat{y}_w) = -[y_1 \log(\hat{y}_1) + \cdots + y_o \log(\hat{y}_o) + \cdots + y_w \log(\hat{y}_w)] \quad = -y_o \log(\hat{y}_o) \; = -\log(\hat{y}_o) \quad = -\log P(O = o|C = c)$$

(b) J is the softmax function of theta which is applied on the output vector to get the predicted probabilities of context vectors.

We want to find the gradients of J wrt to v_c which is the embedded representation of input word(center word in this case).

We can use chain rule to get the gradients related to both input vector here.

(b) we know this deravatives:

$$\because J = CE(y, \hat{y}) \; \hat{y} = softmax(\theta) \; \therefore \frac{\partial J}{\partial \theta} = (\hat{y} - y)^T$$

#http://kb.timniven.com/?p=10 - Check out the derivation here

y is a column vector in the above equation. So, we can use chain rules to solve the deravitive:

$$\frac{\partial J}{\partial v_c} = \frac{\partial J}{\partial \theta}\frac{\partial \theta}{\partial v_c} \quad = (\hat{y} - y)\frac{\partial U^T v_c}{\partial v_c} \; = U^T(\hat{y} - y)^T$$

(c) In the similar way we can apply chain rule to get the gradient wrt to output vector matrix(U).

similar to the equation above.

$$\frac{\partial J}{\partial U} = \frac{\partial J}{\partial \theta}\frac{\partial \theta}{\partial U} \quad = (\hat{y} - y)\frac{\partial U^T v_c}{\partial U} \; = v_c(\hat{y} - y)^T$$

(d)

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

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

$$\frac{\partial \sigma(x)}{\partial x} = \frac{1 \times e^{(-x)}}{(1+e^{-x})^2} = \sigma(x)(1-\sigma(x))$$

$f = g/h$ of x

then
$$\frac{\partial f(x)}{\partial x} = \frac{\frac{\partial g}{\partial x} h(x) - \frac{\partial h}{\partial x} g(x)}{[h(x)]^2}.$$

(e)

(e) As given —

$$J_{neg\text{-}sample} = -\log(\sigma(U_o^T V_c)) - \sum_{k=1}^{K} \log(\sigma(-U_k^T V_c))$$

$$\frac{\partial J_{neg\_sample}}{\partial V_c} = \frac{\sigma'(U_o^T V_c)}{\sigma(U_o^T V_c)} \frac{\partial U_o^T V_c}{\partial V_c} - \sum_{k=1}^{K} \frac{\sigma'(-U_k^T V_c)}{\sigma(-U_k^T V_c)} \frac{\partial(-U_k^T V_c)}{\partial V_c}$$

$$= -(1-\sigma(U_o^T V_c)) V_c + \Sigma (1-\sigma(-U_k^T V_c)) U_k^T$$

[As we know $\sigma'(x) = \sigma(x)(1-\sigma(x))$]

Now
$$\frac{\partial J_{neg\_sample}}{\partial U_o} = -(1-\sigma(U_o^T V_c)) V_c^T$$

and
$$\frac{\partial J_{neg\_sample}}{\partial U_k} = (1-\sigma(-U_k^T V_c)) V_c^T$$

(f)

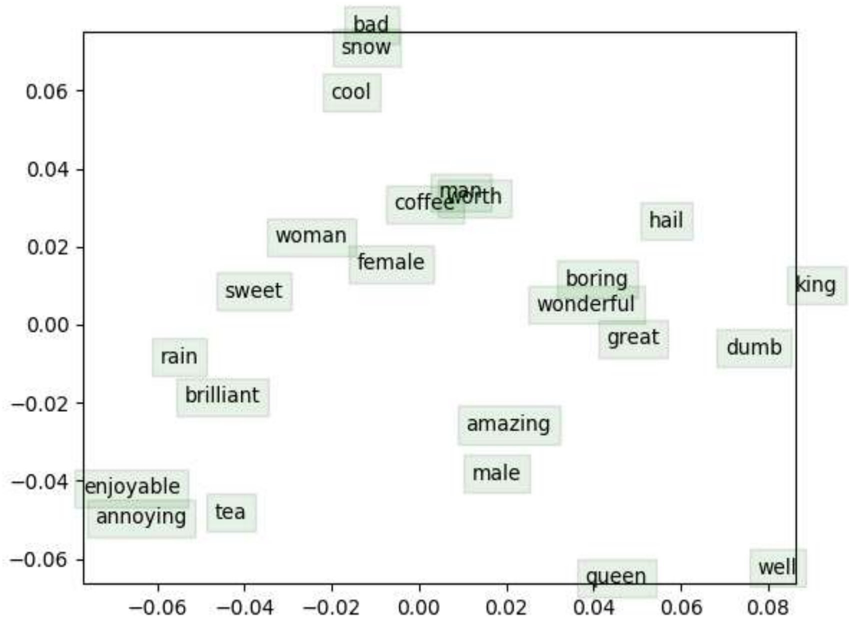
$$\frac{\partial J_{skip\_gram}(\mathbf{v}_c, w_{t-m}, ..., w_{t+m}, \mathbf{U})}{\partial \mathbf{U}} = \sum_{-m \leq j \leq m, j \neq 0} \frac{\partial J(\mathbf{v}_c, w_{t+j}, \mathbf{U})}{\partial \mathbf{U}}$$

$$\frac{\partial J_{skip\_gram}(\mathbf{v}_c, w_{t-m}, ..., w_{t+m}, \mathbf{U})}{\partial \mathbf{v}_c} = \sum_{-m \leq j \leq m, j \neq 0} \frac{\partial J(\mathbf{v}_c, w_{t+j}, \mathbf{U})}{\partial \mathbf{v}_c}$$

$$\frac{\partial J_{skip\_gram}(\mathbf{v}_c, w_{t-m}, ..., w_{t+m}, \mathbf{U})}{\partial \mathbf{v}_w} = 0 \quad when \quad w \neq c$$

Coding Implementation -

c) Last Part of code after only 1000 iterations out of 40000 iterations -



Helpful Links - https://stats.stackexchange.com/questions/253244/gradients-for-skipgram-word2vec

https://courses.cs.ut.ee/MTAT.03.277/2015_fall/uploads/Main/word2vec.pdf

https://deepnotes.io/softmax-crossentropy

https://math.stackexchange.com/questions/945871/derivative-of-softmax-loss-function

http://www.claudiobellei.com/2018/01/06/backprop-word2vec/

https://medium.com/explore-artificial-intelligence/word2vec-a-baby-step-in-deep-learning-but-a-giant-leap-towards-natural-language-processing-40fe4e8602ba

https://towardsdatascience.com/an-implementation-guide-to-word2vec-using-numpy-and-google-sheets-13445eebd281

https://nathanrooy.github.io/posts/2018-03-22/word2vec-from-scratch-with-python-and-numpy/

https://cambridgespark.com/4046-2/

http://kb.timniven.com/?p=10