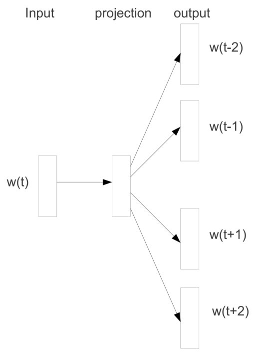
SKIP-GRAM REFRESHER

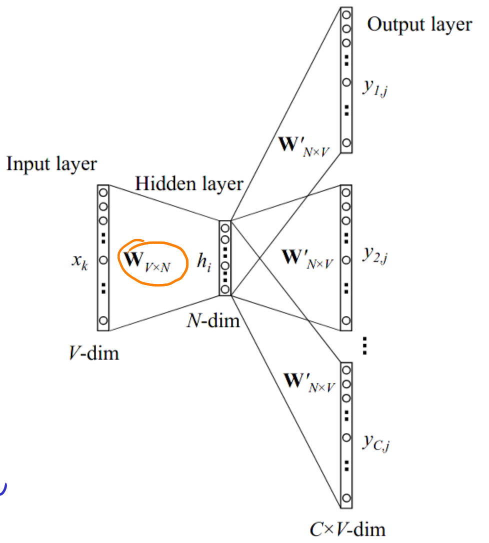


This is the original skip-gram model architecture. ~~It uses a NN with one hidden layer.~~ It receives a one-hot vector as input (center word vector) and outputs a window-size output representing the probabilities of the surrounding context words of the input center word.

The output probabilities are subproducts of the trained model. What we really are interested on are the two matrices learned during training. These matrices are W_{input} and W_{output}

THE SKIP-GRAM NN ARCHITECTURE

- We decide to represent each word with a vector of N dimensions
- We have a corpus of text containing T words and a vocablary of lenght V
- Our corpus of text is represented in two matrices: W_{input}, W_{output}. Each word appears in both matrices. After training we use the W_{input} as our word embeddings
- W_{input} represent each word when it acts as a center word
- W_{output} represents each word when it acts as a context word
- Each vector of W_{input} is called v_c
- Each vector of W_{output} is called u_o
- Each input we feed into the NN is called x_k where k is the position that word occupies in the vocablary. This input is a vector
- During training we select a center word $v_c \to x_k$ and a context window of size C
- The model's output is a set of probabilities y_{c_j}; one per each context word. This probability indicates how likely that word acts as a context word for the input center word.



THE MODEL IN DETAIL

> The man who passes the sentence should swing the sword.
> c_1 v_c c_2
>
> **- Ned Stark**

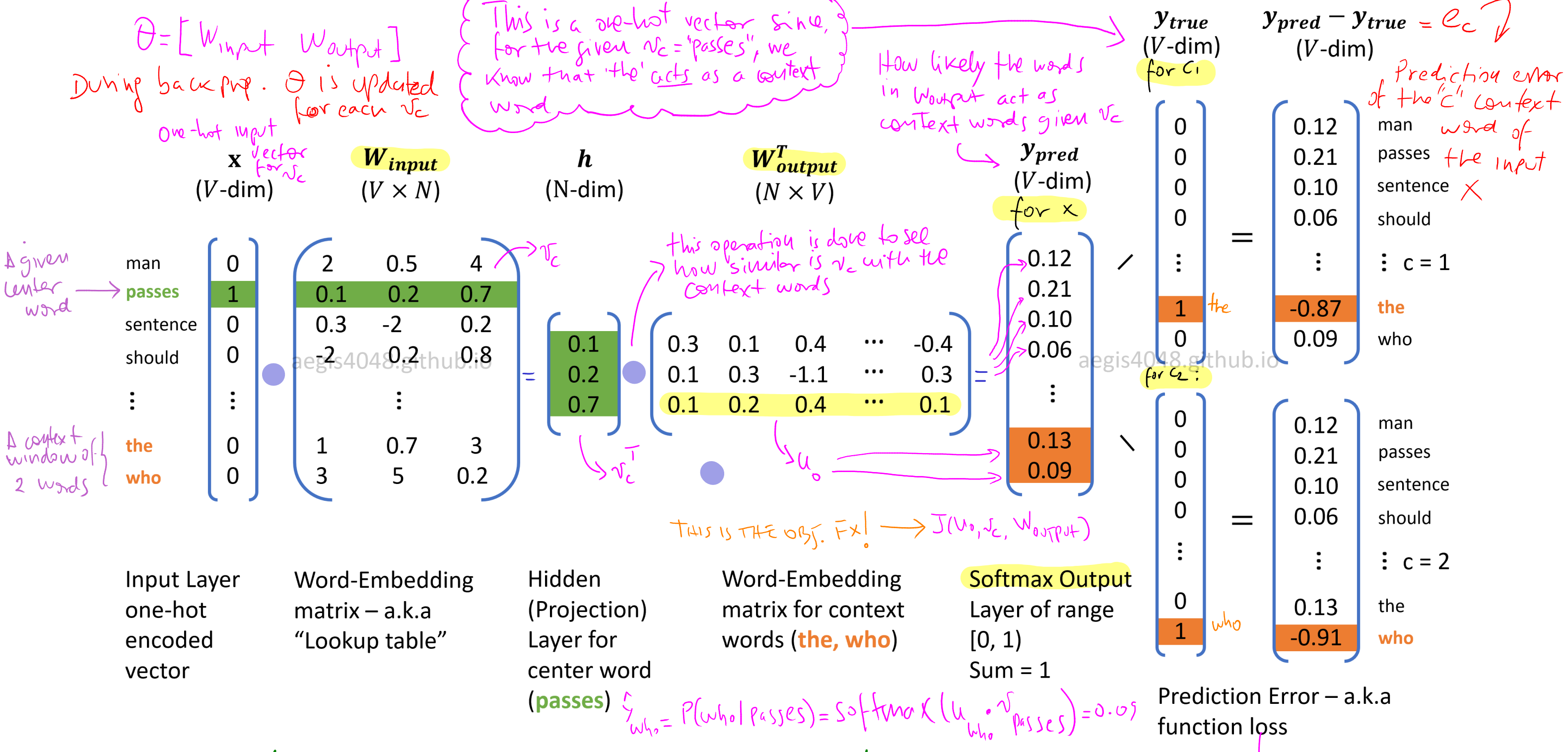
$N=3$ $T=10$ $V=8$ $C=1$

$\theta = [W_{input} \ W_{output}]$

During backprop. θ is updated for each v_c

This is a one-hot vector since, for the given v_c = "passes", we know that 'the' acts as a context word

How likely the words in W_{output} act as context words given v_c



One-hot input vector for v_c — **x** (V-dim)

W_{input} ($V \times N$)

h (N-dim)

W^T_{output} ($N \times V$)

y_{pred} (V-dim) for x

y_{true} (V-dim) for c_1

$y_{pred} - y_{true}$ (V-dim) $= e_c$

Prediction error of the "c" context word of the input X

A given center word → passes

A context window of 2 words

$$\begin{array}{lc} \text{man} & 0 \\ \text{passes} & 1 \\ \text{sentence} & 0 \\ \text{should} & 0 \\ \vdots & \vdots \\ \text{the} & 0 \\ \text{who} & 0 \end{array} \bullet \begin{pmatrix} 2 & 0.5 & 4 \\ 0.1 & 0.2 & 0.7 \\ 0.3 & -2 & 0.2 \\ -2 & 0.2 & 0.8 \\ & \vdots & \\ 1 & 0.7 & 3 \\ 3 & 5 & 0.2 \end{pmatrix} = \begin{pmatrix} 0.1 \\ 0.2 \\ 0.7 \end{pmatrix} \bullet \begin{pmatrix} 0.3 & 0.1 & 0.4 & \cdots & -0.4 \\ 0.1 & 0.3 & -1.1 & \cdots & 0.3 \\ 0.1 & 0.2 & 0.4 & \cdots & 0.1 \end{pmatrix} = \begin{pmatrix} 0.12 \\ 0.21 \\ 0.10 \\ 0.06 \\ \vdots \\ 0.13 \\ 0.09 \end{pmatrix}$$

→ v_c

v_c^T

u_o

this operation is done to see how similar is v_c with the context words

$$\begin{pmatrix} 0.12 \\ 0.21 \\ 0.10 \\ 0.06 \\ \vdots \\ 0.13 \\ 0.09 \end{pmatrix} - \begin{array}{cl} 0 & \\ 0 & \\ 0 & \\ 0 & \\ \vdots & \\ 1 & \text{the} \\ 0 & \end{array} = \begin{array}{cl} 0.12 & \text{man} \\ 0.21 & \text{passes} \\ 0.10 & \text{sentence} \\ 0.06 & \text{should} \\ \vdots & \vdots \ c = 1 \\ -0.87 & \text{the} \\ 0.09 & \text{who} \end{array}$$

for c_2:

$$\begin{pmatrix} 0.12 \\ 0.21 \\ 0.10 \\ 0.06 \\ \vdots \\ 0.13 \\ 0.09 \end{pmatrix} - \begin{array}{cl} 0 & \\ 0 & \\ 0 & \\ 0 & \\ \vdots & \\ 0 & \\ 1 & \text{who} \end{array} = \begin{array}{cl} 0.12 & \text{man} \\ 0.21 & \text{passes} \\ 0.10 & \text{sentence} \\ 0.06 & \text{should} \\ \vdots & \vdots \ c = 2 \\ 0.13 & \text{the} \\ -0.91 & \text{who} \end{array}$$

THIS IS THE OBJ. FX! → $J(u_o, v_c, W_{output})$

Input Layer one-hot encoded vector

Word-Embedding matrix – a.k.a “Lookup table”

Hidden (Projection) Layer for center word (**passes**)

Word-Embedding matrix for context words (**the, who**)

Softmax Output Layer of range [0, 1) Sum = 1

Prediction Error – a.k.a function loss

$\hat{y}_{who} = P(who|passes) = \text{softmax}(u_{who} \cdot v_{passes}) = 0.09$

This is used during backpropagation!

When we feed the NN with a v_c we are also given it the true context words for that word "c". This context words have a Y one-hot true vector for each one. The model outputs $\hat{Y}$ per each context word. The NN is learning to predict the context words, as a byproduct generates the embedings

CS224N – ASSIGNMENT #2

---

a) Demonstrate : For a training pass, given a v_c we have one $\hat{y}$ containing all the probabilities of the vocab words acting as context words!

$$-y_{p_o} \cdot \log \hat{y}_p = -\sum_{w \in V \text{ of } y} \underbrace{y_w \log(\hat{y}_w)}_{\text{not a dot product!}} = -\log(\hat{y}_o) = -\log[P(o|c)] = J(u_o, v_c, U)$$

Word p = "passes"
Context word o = "who"

$w \in V$ of y → Just an index of Vector y

Notice we also have one y for each word in the context window

this happens at index w=o of y

$$-[0 \times \cancel{\log 0.12} + 0 \times \cancel{\log 0.21} + \cancel{\ldots} + 1 \times \log 0.09] = -\log(0.09)$$

b) Compute the partial derivative of $J_{\text{naive-softmax}}(v_c, o, U)$ w.r.t. v_c

$J_{\text{naive-softmax}}$ → Objective function

U → outside matrix (aka W_{output})

o → Current outside word (in the context window)

v_c → Current center word being trained

For a single pair of words the objective function J is :

$$J_{ns}(v_c, o, U) = -\log P(O=o \mid C=c) = -\log P(o|c) = -\log \text{softmax}(u_o v_c) = -\log \frac{e^{u_o^T v_c}}{\sum_{w \in V} e^{u_w^T v_c}} = -\log \hat{y}_o$$

$$J_{ns}(v_c, o, U) = -\sum_{w \in V} y_w \log(\hat{y})$$ when w=o, meaning o is acting as context word of c

$$J_{ns}(v_c, o, U) = -\sum_{w \in V} y_w \log \Bigg[\underbrace{\frac{e^{u_o^T v_c}}{\sum_{x \in V} e^{u_x^T v_c}}}_{P(o|c)} \Bigg] = -\sum_{w \in V} y_w \Big[\log e^{u_o^T v_c} - \log \sum_{x \in V} e^{u_x^T v_c}\Big]$$

We know that when w=o $y_w = 1$ and 0 for other indices given the center word c and context word o we are currently analyzing

$$J_{ns}(v_c, o, U) = -\sum_{w \in V} y_w \Big[u_o^T v_c - \log \sum_{x \in V} e^{u_x^T v_c}\Big]$$

$$J_{ns}(v_c, o, U) = \underbrace{-y_o}_{-1} \Big[u_o^T v_c - \log \sum_{x \in V} e^{u_x^T v_c}\Big]$$

$$U^T \cdot \hat{y}_c - U^T \cdot y_o$$

$$\hat{y}_1 u_1 + \hat{y}_2 u_2 + \ldots \quad U^T(\hat{y}_c - y_o)$$

$$U^T = [[u_1][u_2] \ldots [u_v]] \quad [\;]^{\hat{y}_c}$$

$$\frac{\partial J_{ns}}{\partial v_c} = -\frac{\partial}{\partial v_c} u_o^T v_c + \frac{\partial}{\partial v_c} \log \sum_{x \in V} e^{u_x^T v_c}$$

$\frac{\partial J_{ns}}{\partial v_c}$ → Shape is 1xN

$$= -u_o + \log e \times \frac{1}{\sum_{x \in V} e^{u_x^T v_c}} \times \frac{\partial}{\partial v_c} \sum_{x \in V} e^{u_x^T v_c}$$

Not sure about moving this sum out!

"virguerías" start here

$$= -u_o + \frac{\sum_{x \in V} \partial/\partial v_c \, e^{u_x^T v_c}}{\sum_{x \in V} e^{u_x^T v_c}} = -u_o + \sum_{x \in V} \frac{e^{u_x^T v_c}}{\sum_{j \in V} e^{u_j^T v_c}} u_x = -u_o + \underbrace{\sum_{x \in V} P(x|c) u_x}_{\text{Shape is vector of 1xN}} = -u_o + \sum_{x \in V} \hat{y}_x u_x$$

Shape is vector of 1xN ⇒ follows the shape convention

current context word o has an y_o

$$= \sum_{x \in V} \underbrace{\hat{y}_x u_x}_{\text{Dot product!}} - u_o = U^T \hat{y} - U^T y_o = U^T(\hat{y} - y_o)$$

$$\hat{y}_1 \cdot u_1 + \hat{y}_2 \cdot u_2 + \ldots + \hat{y}_v \cdot u_v = U^T \cdot \hat{y}$$

↳ vectors when processing word 2

ALTERNATIVE CALCULATION:

$$\frac{\partial}{\partial v_c} -\log \frac{e^{u_o^T v_c}}{\sum_{w \in V} e^{u_w^T v_c}} = -\frac{\partial}{\partial v_c}\left[\log e^{u_o^T v_c} - \log \sum_{w \in V} e^{u_w^T v_c}\right]$$

composite function

$$= \underbrace{-\frac{\partial}{\partial v_c} \log \overbrace{e^{u_o^T v_c}}}_{} + \left.\frac{\partial}{\partial v_c} \log \sum_{w_c \in V} e^{u_w^T v_c}\right\}$$

② See next page

?
$\log_e e = 1$
$\log e = 0.43$
probably log in this context is ln

$$= -\log e \times \frac{1}{e^{u_o^T v_c}} \times \frac{\partial}{\partial v_c} e^{u_o^T v_c}$$

$$= -\frac{1}{\cancel{e^{u_o^T v_c}}} \times \cancel{e^{u_o^T v_c}} \times \frac{\partial u_o^T v_c}{\partial v_c}$$

$$= -u_o$$

$u_o = [u_{o1}\ u_{o2} \cdots u_{oV}]$

$u_o^T v_c = u_{o1} v_{c_1} + u_{o2} v_{c_2} \ldots u_{oV} v_{c_V}$

$\frac{\partial}{\partial v_{c_1}} u_o^T v_c = \underline{\underline{u_{o_1}}} + 0 + 0 \ldots + 0$

$\frac{\partial}{\partial v_{c_2}} u_o^T v_c = 0 + \underline{\underline{u_{o_2}}} + 0 \ldots + 0$

$\frac{\partial}{\partial v_{c_3}} u_o^T v_c = 0 + 0 + \underline{\underline{u_{o_3}}} + 0 + \ldots + 0$

Partial derivatives of v_c

$\Rightarrow \frac{\partial}{\partial v_c} u_o^T v_c = u_o$

Recall: Ⓐ
If $F(x) = \frac{f(x)}{g(x)} \Rightarrow -\log F(x) = -\log f(x) + \log g(x)$

Recall: if s is a function of x; and Ⓑ $y = \log_b s$; then $\frac{dy}{dx} = (\log_b e)\frac{s'}{s}$

Recall Ⓒ: If u is a function of x; $\frac{d}{dx} e^u = e^u \frac{du}{dx}$

↳ This is chain rule!

$$② \frac{\partial}{\partial v_c} \underbrace{\log}_{\text{composite}} \sum_{w_c \in V} \underbrace{e^{u_w^T v_c}}_{\text{A function of } v_c} = \log e \times \frac{1}{\sum_{w \in V} e^{u_w^T v_c}} \times \frac{\partial}{\partial v_c} \sum_{x \in V} e^{u_x^T v_c}$$

Apply Ⓑ

It is needed to change variables! why? Not sure

$$\log e \times \frac{1}{\sum_{w \in V} e^{u_w^T v_c}} \times \sum_{x \in V} \frac{\partial}{\partial v_c} e^{u_x^T v_c}$$

I guess just for the softmax

$$\underline{\log e} \times \frac{\sum_{x \in V} e^{u_x^T v_c} u_x}{\sum_{w \in V} e^{u_w^T v_c}}$$

Some people uses log for ln; it seems this is the case here

$$\sum_{x \in V} \frac{e^{u_x^T v_c}}{\sum_{w \in V} e^{U_w^T v_c}} u_x$$

observed representation of the context word

$$\frac{\partial J(v_c, o, U)}{\partial v_c} = \underbrace{-u_o}_{\text{Part 1}} + \underbrace{\sum_{x \in V} P(x|c)\, u_x}_{\text{The expected context word}} = -u_o + \sum_{x \in V} \hat{y}_x u_x = -u_o + U^T \hat{y}$$

⟹ what our model thinks what the context should look like

One-hot vector!

$$\underbrace{\frac{\partial J(v_c, o, U)}{\partial v_c}}_{\text{a slope in the multidimensional space from the}} = \underbrace{-y + U^T \hat{y}}_{\text{slope meaning}} = \underbrace{-U^T y + U^T \hat{y}}_{\text{The shape of this is } 1 \times N \text{, the same of } v_c} = U^T (\hat{y} - y) \,//\!/.$$

$$U = \begin{bmatrix} u_0 & u_1 & \cdots & u_V \\ \vdots & \vdots & & \vdots \end{bmatrix}_{d \times V}$$ } d-dim.

$y = \begin{bmatrix} \vdots \end{bmatrix}$ $\hat{y} = \begin{bmatrix} \vdots \end{bmatrix}$

for a given center word c, y is a hot vector with 1 in the k position where that word is a true outside word
T

a row in matrix U (W_{output})

This means when $u_w = u_o$ => one of the context words of the given center word c

(c) (5 points) Compute the partial derivatives of $\boldsymbol{J}_{\text{naive-softmax}}(\boldsymbol{v}_c, o, \boldsymbol{U})$ with respect to each of the 'outside' word vectors, $\boldsymbol{u}_w$'s. There will be two cases: when $w = o$, the true 'outside' word vector, and $w \neq o$, for all other words. Please write your answer in terms of $\boldsymbol{y}$, $\hat{\boldsymbol{y}}$, and $\boldsymbol{v}_c$. In this subpart, you may use specific elements within these terms as well, such as $(\boldsymbol{y}_1, \boldsymbol{y}_2, \dots)$.

$$\frac{\partial}{\partial u_w} J_{ns}(v_c, o, U) = \frac{\partial}{\partial u_w} -\log \frac{e^{u_o^T v_c}}{\sum_{w \in V} e^{u_w^T v_c}} = \frac{\partial}{\partial u_w} -\left[\log e^{u_o^T v_c} - \log \sum_{w \in V} e^{u_w^T v_c}\right]$$

∂u_w → shape is 1xN

When $u_{w \neq o}$:

For the given center word "c", any word in $U(W_{output})$ that is not acting as context word

$$= \underbrace{-\frac{\partial}{\partial u_w} \log e^{u_o^T v_c}}_{0} + \frac{\partial}{\partial u_w} \log \sum_{w \in V} e^{u_w^T v_c}$$

→ constant

$$= 0 + \log e \times \frac{1}{\sum e^{u_w^T v_c}} \times \frac{\partial \sum_{w \in V} e^{u_w^T v_c}}{\partial u_w}$$

Chain rule!

$$= \underbrace{\frac{\sum_{w \in V} e^{u_w^T v_c}}{\sum_{x \in V} e^{u_x^T v_c}}} v_c$$

As u_w.shape = 1xN => Because of shape convention $\frac{\partial J}{\partial u_w}$ should also has a shape equals to 1xN.

Why does the sum dissapears?

↳ virguerías !! :)

Perhaps cause we have a fixed word w and c

$$= \sum_{w \in V} P(w|c) v_c$$

$$= \hat{y}_{w \neq o} v_c = \underbrace{\hat{y}_{w \neq o}}_{\text{scalar}} v_c - 0 \approx \hat{y}_{w \neq o} v_c - \underbrace{y_w v_c}_{\text{Not sure about this}} = v_c(\hat{y}_{w \neq o} - \underbrace{y_w}_{\text{scalar}})$$

just an index

as w is not a context word, the true vector is zero

→ vector index

When $u_{w=o}$: for the outside matrix vectors corresponding to the context words of v_c in the context window

$$\frac{\partial}{\partial u_o} J_{ns}(v_c, o, U) = -\frac{\partial}{\partial u_o} \log e^{u_o^T v_c} + \frac{\partial}{\partial u_o} \log \sum_{o \in V} e^{u_o^T v_c}$$

chain rule!

$$= -\log e \times \frac{1}{e^{u_o^T v_c}} \times \frac{\partial}{\partial u_o} e^{u_o^T v_c} + \log e \times \frac{1}{\sum_{o \in V} e^{u_o^T v_c}} \times \frac{\partial}{\partial u_o} \sum_{o \in V} e^{u_o^T v_c}$$

$$= -\frac{\cancel{e^{u_o^T v_c}} v_c}{\cancel{e^{u_o^T v_c}}} + \frac{\sum_{o \in V} e^{u_o^T v_c} v_c}{\sum_{w \in V} e^{u_w^T v_c}}$$

$$= -v_c + \sum_{o \in V} \underbrace{P(o|c)} v_c$$

The resulting value is a vector 1xN, same shape of u_o

$$= -v_c + \sum_{o \in V} \hat{y}_o v_c$$

↳ This is just an index! $\equiv \hat{y}_{w=o}[0]$

$$= v_c(\hat{y}_o - 1) = v_c(\hat{y}_o - y_o)$$

$\frac{\partial}{\partial U} J$ is a matrix!

(d) (1 point) Compute the partial derivative of $\boldsymbol{J}_{\text{naive-softmax}}(\boldsymbol{v}_c, o, \boldsymbol{U})$ with respect to U. Please write your answer in terms of $\frac{\partial \boldsymbol{J}(\boldsymbol{v}_c,o,\boldsymbol{U})}{\partial \boldsymbol{u}_1}, \frac{\partial \boldsymbol{J}(\boldsymbol{v}_c,o,\boldsymbol{U})}{\partial \boldsymbol{u}_2}, \ldots, \frac{\partial \boldsymbol{J}(\boldsymbol{v}_c,o,\boldsymbol{U})}{\partial \boldsymbol{u}_{|Vocab|}}$. The solution should be one or two lines long.

$$\frac{\partial}{\partial U} J(v_c, o, U) = \left[\frac{\partial}{\partial u_1} J \quad \frac{\partial}{\partial u_2} J \cdots \frac{\partial}{\partial u_o} J \cdots \frac{\partial}{\partial u_v} J\right] = \left[\hat{y}_{u_1} v_c \quad \hat{y}_{u_1} v_c \cdots (\hat{y}_o - 1) v_c \cdots \hat{y}_{u_v} v_c\right]$$

↳ The result should be a matrix!

(e) (3 Points) The sigmoid function is given by Equation 4:

$$\sigma(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1} \tag{4}$$

Please compute the derivative of $\sigma(x)$ with respect to x, where x is a scalar. Hint: you may want to write your answer in terms of $\sigma(x)$.

$$\sigma(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1}$$ → I'm going to choose this

$$\frac{\partial}{\partial x} \sigma(x) = \frac{(e^x + 1)e^x - e^x(\partial/\partial x\, e^x + 1)}{(e^x + 1)^2}$$ { Quotient rule

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

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

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

$$= \sigma(x) \frac{e^x + 1}{e^x + 1} - \sigma(x)^2 = \sigma(x)\left[\frac{e^x + 1}{e^x + 1} - \sigma(x)\right] = \sigma(x)\left[1 - \sigma(x)\right]$$

(f) (4 points) Now we shall consider the Negative Sampling loss, which is an alternative to the Naive Softmax loss. Assume that K negative samples (words) are drawn from the vocabulary. For simplicity of notation we shall refer to them as $w_1, w_2, \ldots, w_K$ and their outside vectors as $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_K$. For this question, assume that the K negative samples are distinct. In other words, $i \neq j$ implies $w_i \neq w_j$ for $i, j \in \{1, \ldots, K\}$. Note that $o \notin \{w_1, \ldots, w_K\}$. For a center word c and an outside word o, the negative sampling loss function is given by:

$$\boldsymbol{J}_{\text{neg-sample}}(\boldsymbol{v}_c, o, \boldsymbol{U}) = -\log(\sigma(\boldsymbol{u}_o^\top \boldsymbol{v}_c)) - \sum_{k=1}^{K} \log(\sigma(-\boldsymbol{u}_k^\top \boldsymbol{v}_c)) \tag{5}$$

for a sample $w_1, \ldots w_K$, where $\sigma(\cdot)$ is the sigmoid function.[4]

Please repeat parts (b) and (c), computing the partial derivatives of $\boldsymbol{J}_{\text{neg-sample}}$ with respect to $\boldsymbol{v}_c$, with respect to $\boldsymbol{u}_o$, and with respect to a negative sample $\boldsymbol{u}_k$. Please write your answers in terms of the vectors $\boldsymbol{u}_o$, $\boldsymbol{v}_c$, and $\boldsymbol{u}_k$, where $k \in [1, K]$. After you've done this, describe with one sentence why this loss function is much more efficient to compute than the naive-softmax loss. Note, you should be able to use your solution to part (e) to help compute the necessary gradients here.

Current → outside word in context of v_c

K → K negative samples (words). All are distinct

$$J_{\text{neg-sample}}(v_c, o, U) = -\log\left[\sigma(u_o^T v_c)\right] - \sum_{k=1}^{K} \log\left[\sigma(-u_k^T v_c)\right]$$

v_c → given center word

U → outside Matrix

u_k → outside vectors; $o \notin \{w_1 \cdots w_K\}$

$u_k \neq u_o$

Note: In numpy $\text{np.dot}(-u_k, v_c) \equiv \text{np.dot}(-u_k^T, v_c)$

①
$$\frac{\partial}{\partial v_c} J = -\left\{ \log e \times \frac{1}{\sigma(u_o^T v_c)} \times \underbrace{\frac{\partial}{\partial v_c} \sigma(u_o^T v_c)}_{\text{chain rule!}} \right\} - \sum_{k \in K} \left\{ \log e \cdot \frac{1}{\sigma(-u_k^T v_c)} \times \frac{\partial}{\partial v_c} \sigma(-u_k^T v_c) \right\}$$

$$= -\left\{ \log e \times \frac{1}{\sigma(u_o^T v_c)} \times \frac{\partial}{\partial v_c} \sigma(u_o^T v_c) \times \frac{\partial}{\partial v_c} u_o^T v_c \right\} - \sum \left\{ \log e \cdot \frac{1}{\sigma(-u_k^T v_c)} \times \frac{\partial}{\partial v_c} \sigma(-u_k^T v_c) \cdot \frac{\partial}{\partial v_c} -u_k^T v_c \right\}$$

$$= -\left\{ \log e \times \frac{1}{\sigma(\cancel{u_o^T v_c})} \times \sigma(\cancel{u_o^T v_c}) \left[1 - \sigma(u_o^T v_c)\right] \times u_o \right\} - \sum \left\{ \log e \times \frac{1}{\sigma(\cancel{-u_k^T v_c})} \times \sigma(\cancel{-u_k^T v_c}) \left[1 - \sigma(-u_k^T v_c)\right] \times -u_k \right\}$$

$$\frac{\partial}{\partial v_c} J_{nes} = -u_o \left[1 - \sigma(u_o^T v_c)\right] + \sum_{k \in K} u_k \left[1 - \sigma(-u_k v_c)\right] = \boxed{u_o \left[\sigma(u_o^T v_c) - 1\right] - \sum_{k \in K} u_k \left[\sigma(-u_k^T v_c) - 1\right]}$$

②
$$\frac{\partial}{\partial u_o} J_{nes} = -\left\{ \log e \times \frac{1}{\sigma(u_o^T v_c)} \times \frac{\partial}{\partial u_o} \sigma(u_o^T v_c) \right\} - \sum_{k \in K} \frac{\partial}{\partial u_o} \underbrace{\log\left[\sigma(-u_k^T v_c)\right]}_{\text{scalar}}$$

σ is a function of u_o then chain rule!

$$= -\left\{ \frac{1}{\sigma(u_o^T v_c)} \times \frac{\partial}{\partial u_o} \sigma(u_o^T v_c) \times \frac{\partial}{\partial u_o} u_o^T v_c \right\} - 0$$

$$= -\left\{ \frac{1}{\sigma(\cancel{u_o^T v_c})} \times \sigma(\cancel{u_o^T v_c}) \left[1 - \sigma(u_o^T v_c\right] \times v_c \right\}$$

$$= -v_c \left[1 - \sigma(u_o^T v_c)\right] = \boxed{v_c \left[\sigma(u_o^T v_c) - 1\right]}$$

③
$$\frac{\partial}{\partial u_k} J_{nes} = -\left\{ 0 \right\} - \sum \frac{\partial}{\partial u_k} \log\left[\sigma(-u_k^T v_c)\right]$$

$$= -\sum \log e \times \frac{1}{\sigma(-u_k^T v_c)} \times \frac{\partial}{\partial u_k} \sigma(-u_k^T v_c)$$

chain rule!

$$= -\sum \frac{1}{\sigma(\cancel{-u_k^T v_c})} \times \sigma(\cancel{-u_k^T v_c}) \left[1 - \sigma(-u_k^T v_c)\right] \times -v_c$$

$$= -\sum_{k \in K} -v_c \left[1 - \sigma(-u_k^T v_c)\right] = -\sum_{k \in K} v_c \left[\sigma(-u_k^T v_c) - 1\right]$$

OR $\longrightarrow$ $-v_c \left[\sigma(-u_k^T v_c) - 1\right], \ \forall k \in [1, K]$

↳ This notation is useful when inside a for loop of the k-samples.

Why this loss function is much more efficient to compute than the naive soft-max loss?

⇒ With the naive soft-max we need to normalize the probability by multiplying all the vocab vectors with the center word. Here we do it just K times.

h) Recall the skip-gram version of word2vec, the total loss for the context window is:

$$J_{\text{skip-gram}}(v_c, w_{t-m}, \dots, w_{t+m}, U) = \sum_{\substack{-m \le j \le m \\ j \ne 0}} J \underbrace{(v_c, \overset{\text{outside word}}{w_{t+j}}, U)}_{\text{Arbitrary loss term for the center word } c = w_t} = \sum_{\substack{-m \le j \le m \\ j \ne 0}} -\log P(w_{t+j} | w_t)$$

We can attach here naive soft-max or neg-sampling

1) $$\frac{\partial}{\partial v_c} J_{SG}(v_c, w_{t+j}, U) = \sum_{\substack{-m \le j \le m \\ j \ne 0}} \frac{\partial}{\partial v_c} J(v_c, w_{t+j}, U)$$

2) $$\frac{\partial}{\partial v_w} J_{SG}(v_c, w_{t+j}, U) = \sum_{\substack{-m \le j \le m \\ j \ne 0}} \frac{\partial}{\partial v_{w \ne c}} J(v_c, w_{t+j}, U) = 0$$; J is not a function of v_w

Probably they refer to:

$$\frac{\partial}{\partial u_{w \ne c}} J_{SG}(v_c, w_{t+j}, U) = \sum_{\substack{-m \le j \le m \\ j \ne 0}} \frac{\partial}{\partial u_w} J(v_c, w_{t+j}, U)$$

3) $$\frac{\partial}{\partial U} J_{SG}(v_c, w_{t+j}, U) = \sum_{\substack{-m \le j \le m \\ j \ne 0}} \frac{\partial}{\partial U} J(v_c, w_{t+1}, U)$$