$$P(O = o \mid C = c) = \frac{\exp(\boldsymbol{u}_o^\top \boldsymbol{v}_c)}{\sum_{w \in \text{Vocab}} \exp(\boldsymbol{u}_w^\top \boldsymbol{v}_c)} \quad (2) \qquad J_{\text{naive-softmax}}(\boldsymbol{v}_c, o, \boldsymbol{U}) = -\log P(O = o | C = c). \quad (1)$$

(a) (2 points) Prove that the naive-softmax loss (Equation 2) is the same as the cross-entropy loss between $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$, i.e. (note that $\boldsymbol{y}, \hat{\boldsymbol{y}}$ are vectors and $\hat{\boldsymbol{y}}_o$ is a scalar):

$$-\sum_{w \in \text{Vocab}} \boldsymbol{y}_w \log(\hat{\boldsymbol{y}}_w) = -\log(\hat{\boldsymbol{y}}_o). \quad (3)$$

Your answer should be one line. You may describe your answer in words.

(a) $\sum_{w \in \text{Vocab}} y_w \log(\hat{y}_w) = \sum_{w \in O} y_w \log(\hat{y}_w) = \sum_{w \in O} \log \hat{y}_w = \log \hat{y}_o$

$y_w = 1 \quad w \in O$
$y_w = 0 \quad w \notin O$
$y$ is one hot encoder

$y_w = 1$ $\quad w \in O$

(b) (7 points)

(i) Compute the partial derivative of $J_{\text{naive-softmax}}(\boldsymbol{v}_c, o, \boldsymbol{U})$ with respect to $\boldsymbol{v}_c$. Please write your answer in terms of $\boldsymbol{y}$, $\hat{\boldsymbol{y}}$, $\boldsymbol{U}$, and show your work to receive full credit.

- **Note**: Your final answers for the partial derivative should follow the shape convention: the partial derivative of any function $f(x)$ with respect to $x$ should have the **same shape** as $x$.[4]
- Please provide your answers for the partial derivative in vectorized form. For example, when we ask you to write your answers in terms of $\boldsymbol{y}$, $\hat{\boldsymbol{y}}$, and $\boldsymbol{U}$, you may not refer to specific elements of these terms in your final answer (such as $\boldsymbol{y}_1$, $\boldsymbol{y}_2$, ...).

(b)(i) $\frac{\partial J}{\partial v_c} = \frac{\partial}{\partial v_c} - \log P(O=o|C=c) = \frac{\partial}{\partial v_c} - \log \frac{e^{u_o^T v_c}}{\sum_{w \in V} e^{u_w^T v_c}} =$ (1) (2)

$\log \frac{x}{y} = \log x - \log y$

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

$\frac{\partial \log f(x)}{\partial x} = \frac{\frac{\partial f(x)}{\partial x}}{f(x)}$

$\frac{\sum_{w \in V} u_w \cdot e^{u_w^T v_c}}{\sum_{w \in V} e^{u_w^T v_c}} - u_o^T = \sum_{w \in V} u_w P(O=w|C=c) - u_o^T =$ (2)

$\hat{y}$

$U^T \cdot \hat{y} - u_o^T = U^T (\hat{y} - y)$

$y = 0 \quad w \notin O$
$y = 1 \quad w \in O$

(ii) When is the gradient you computed equal to zero?
**Hint:** You may wish to review and use some introductory linear algebra concepts.

(iii) The gradient you found is the difference between two terms. Provide an interpretation of how each of these terms improves the word vector when this gradient is subtracted from the word vector $v_c$.

(iv) In many downstream applications using word embeddings, L2 normalized vectors (e.g. $\mathbf{u}/||\mathbf{u}||_2$ where $||\mathbf{u}||_2 = \sqrt{\sum_i u_i^2}$) are used instead of their raw forms (e.g. $\mathbf{u}$). Now, suppose you would like to classify phrases as being positive or negative. When would L2 normalization take away useful information for the downstream task? When would it not? Hint: Consider the case where $\mathbf{u}_x = \alpha \mathbf{u}_y$ for some words $x \neq y$ and some scalar $\alpha$.

(ii) when there are no words that are embedded or when it classifies perfectly the outside words and there are no other words that are predicted for the center word. When y hat = y

(iii) the first term is only the outside words that are related to the center word , adding it will help the gradient to keep the relevant words with high score. The second term is the probability for each word in the vocabulary will be an outside word for the center word, subtracting it will push away the non outside words.

(iv) $U_x = \alpha U_y$ When L2 normalization equals $U_x = U_y$

$$\frac{\alpha U_x}{\sqrt{\sum \alpha^2 U_x^2}} = \frac{\alpha U_x}{\alpha \sqrt{\sum U_x^2}} = \frac{\alpha U_x}{\alpha \sqrt{\sum U_x^2}}$$

(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 $\boldsymbol{y}_1$, $\boldsymbol{y}_2$, ...). Note that $\boldsymbol{u}_w$ is a vector while $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots$ are scalars. Show your work to receive full credit.

$$(c)\frac{\partial J}{\partial U_w} = \frac{\partial}{\partial U_w}\left(\log \sum_{w \in V} e^{U_w^T V_c} - U_o^T V_c\right) =$$

$$\frac{\sum_{w \in V} V_c \cdot e^{U_w V_c}}{\sum_{w \in V} e^{U_w V_c}} \cdot \frac{\partial U_w V_c}{\partial U_w} = \sum_{w \in V} V_c \cdot P(O=w|C=c) - \frac{\partial U_o V_c}{\partial U_w} =$$

$v_c \cdot \hat{y}_w - v_c \cdot \frac{\partial v_o}{\partial v_w} = v_c \cdot \left(\hat{y}_w - \frac{\partial v_o}{\partial v_w}\right) =$

$v_c(\hat{y}_w - 1)$ when $w = o$

$= v_c(\hat{y}_w - y_w)$

$v_c \hat{y}_w$ when $w \neq o$ $y$ is one hot encoder

(d) (1 point) Write down the partial derivative of $\boldsymbol{J}_{\text{naive-softmax}}(\boldsymbol{v}_c, o, \boldsymbol{U})$ with respect to $\boldsymbol{U}$. Please break down your answer in terms of the column vectors $\frac{\partial \boldsymbol{J}(\boldsymbol{v}_c, o, \boldsymbol{U})}{\partial \boldsymbol{u}_1}, \frac{\partial \boldsymbol{J}(\boldsymbol{v}_c, o, \boldsymbol{U})}{\partial \boldsymbol{u}_2}, \ldots, \frac{\partial \boldsymbol{J}(\boldsymbol{v}_c, o, \boldsymbol{U})}{\partial \boldsymbol{u}_{|\text{Vocab}|}}$. No derivations are necessary, just an answer in the form of a matrix.

$\frac{\partial J}{\partial v_w} = v_c(\hat{y}_w - y_w)$ so $\frac{\partial J}{\partial U} = v_c(\hat{y} - y) =$

$\left[\frac{\partial J}{\partial v_1}, \frac{\partial J}{\partial v_2}, \ldots, \frac{\partial J}{\partial v_w}\right]$

(e) (2 points) The Leaky ReLU (Leaky Rectified Linear Unit) activation function is given by Equation 4 and Figure 2:

$$f(x) = \max(\alpha x, x) \tag{4}$$

Where $x$ is a scalar and $0 < \alpha < 1$, please compute the derivative of $f(x)$ with respect to $x$. You may ignore the case where the derivative is not defined at 0.[5]

$\frac{\partial \max(\alpha x, x)}{\partial x} = \frac{\partial \{x \text{ when } x > 0;\ \alpha x \text{ when } x < 0\}}{\partial x} = \begin{cases} 1 & \text{when } x > 0 \\ \alpha & \text{when } x < 0 \end{cases}$

$$\sigma(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1} \tag{5}$$

Please compute the derivative of $\sigma(x)$ with respect to $x$, where $x$ is a scalar. Please write your answer in terms of $\sigma(x)$. Show your work to receive full credit.

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

(5)

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

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

(g) (6 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 $\boldsymbol{u}_{w_1}, \boldsymbol{u}_{w_2}, \ldots, \boldsymbol{u}_{w_K}$.[6] 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:

$$\boldsymbol{J}_{\text{neg-sample}}(\boldsymbol{v}_c, o, \boldsymbol{U}) = -\log(\sigma(\boldsymbol{u}_o^\top \boldsymbol{v}_c)) - \sum_{s=1}^{K} \log(\sigma(-\boldsymbol{u}_{w_s}^\top \boldsymbol{v}_c)) \tag{6}$$

for a sample $w_1, \ldots w_K$, where $\sigma(\cdot)$ is the sigmoid function.[7]

(i) 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 the $s^{th}$ negative sample $\boldsymbol{u}_{w_s}$. Please write your answers in terms of the vectors $\boldsymbol{v}_c$, $\boldsymbol{u}_o$, and $\boldsymbol{u}_{w_s}$, where $s \in [1, K]$. Show your work to receive full credit. **Note:** you should be able to use your solution to part (f) to help compute the necessary gradients here.

$$\frac{\partial J}{\partial V_c} = \frac{\partial -\log\sigma(U_o^T V_c)}{\partial V_c} - \frac{\partial \sum_{s=1}^{K} \log\sigma(-U_{w_s}^T V_c)}{\partial V_c}$$

$$= -\frac{\sigma(U_o^T V_c)\cdot(1-\sigma(U_o^T V_c))\,U_o}{\sigma(U_o^T V_c)} + \sum_{s=1}^{K} \frac{\sigma(-U_{w_s}^T V_c)\left(1-\sigma(-U_{w_s}^T V_c)\right)U_{w_s}}{\sigma(-U_{w_s}^T V_c)}$$

$$= (\sigma(U_o^T V_c) - 1)\,U_o + \sum_{s=1}^{K}\left[1-\sigma(-U_{w_s}^T V_c)\right]U_{w_s}$$

$$\frac{\partial J}{\partial U_o} = \frac{\partial -\log \sigma(U_o^T V_c)}{\partial U_o} - \frac{\partial \sum_{s=1}^{K} \log \sigma(-U_{w_s}^T V_c)}{\partial U_o} =$$

$$-\frac{\sigma(U_o^T V_c)(1-\sigma(U_o^T V_c)) \cdot V_c}{\sigma(U_o^T V_c)} = V_c \cdot (\sigma(U_o^T V_c - 1)$$

$$\frac{\partial J}{\partial U_{w_s}} = \frac{\partial -\log \sigma(U_o^T V_c)}{\partial U_{w_s}} - \frac{\partial \sum_{s=1}^{K} \log \sigma(-U_{w_s}^T V_c)}{\partial U_{w_s}} =$$

$$\sum_{s=1}^{K} \frac{\sigma(-U_{w_s} V_c) \cdot (1-\sigma(-U_{w_s} V_c)) \cdot V_c}{\sigma(-U_{w_s} V_c)} =$$

$$\sum_{s=1}^{K} V_c \cdot [1 - \sigma(-U_{w_s} V_c)]$$

(ii) In lecture, we learned that an efficient implementation of backpropagation leverages the re-use of previously-computed partial derivatives. Which quantity could you reuse amongst the three partial derivatives calculated above to minimize duplicate computation? Write your answer in terms of

$\boldsymbol{U}_{o,\{w_1,\ldots,w_K\}} = [\boldsymbol{u}_o, -\boldsymbol{u}_{w_1}, \ldots, -\boldsymbol{u}_{w_K}]$, a matrix with the outside vectors stacked as columns, and $\mathbf{1}$, a $(K+1) \times 1$ vector of 1's.[8] Additional terms and functions (other than $\boldsymbol{U}_{o,\{w_1,\ldots,w_K\}}$ and $\mathbf{1}$) can be used in your solution.

$$\sigma(U_o^T V_c) - 1 \quad , 1 - \sigma(-U_{w_s} V_c)$$

(iii) Describe with one sentence why this loss function is much more efficient to compute than the naive-softmax loss.

(iii) it takes small number of k instead all of the vocabulary.

(h) (2 points) Now we will repeat the previous exercise, but without the assumption that the $K$ sampled words are distinct. 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 $\boldsymbol{u}_{w_1}, \ldots, \boldsymbol{u}_{w_K}$. In this question, you may not assume that the words are distinct. In other words, $w_i = w_j$ may be true when $i \neq j$ is true. 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:

$$\boldsymbol{J}_{\text{neg-sample}}(\boldsymbol{v}_c, o, \boldsymbol{U}) = -\log(\sigma(\boldsymbol{u}_o^\top \boldsymbol{v}_c)) - \sum_{s=1}^{K} \log(\sigma(-\boldsymbol{u}_{w_s}^\top \boldsymbol{v}_c)) \tag{7}$$

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

Compute the partial derivative of $\boldsymbol{J}_{\text{neg-sample}}$ with respect to a negative sample $\boldsymbol{u}_{w_s}$. Please write your answers in terms of the vectors $\boldsymbol{v}_c$ and $\boldsymbol{u}_{w_s}$, where $s \in [1, K]$. Show your work to receive full credit. Hint: break up the sum in the loss function into two sums: a sum over all sampled words equal to $w_s$ and a sum over all sampled words not equal to $w_s$. Notation-wise, you may write 'equal' and 'not equal' conditions below the summation symbols, such as in Equation 8.

$$\frac{\partial J}{\partial U_{w_s}} = \frac{\partial}{\partial U_{w_s}} \sum_{s=1}^{K} \log \sigma(-U_{w_s}^T V_c) =$$

$$-\frac{\partial}{\partial U_{w_s}} \sum_{w_j = w_s} \log \sigma(-U_{w_j}^T V_c) + \sum_{w_j \neq w_s} \log(-U_{w_j}^T V_c) =$$

$$-\sum_{w_j = w_s} \left[\sigma(-U_{w_j}^T V_c) - 1\right] V_c$$

(i) (3 points) Suppose the center word is $c = w_t$ and the context window is $[w_{t-m}, \ldots, w_{t-1}, w_t, w_{t+1}, \ldots, w_{t+m}]$, where $m$ is the context window size. Recall that for the skip-gram version of `word2vec`, the total loss for the context window is:

$$\boldsymbol{J}_{\text{skip-gram}}(\boldsymbol{v}_c, w_{t-m}, \ldots w_{t+m}, \boldsymbol{U}) = \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \boldsymbol{J}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U}) \tag{8}$$

Here, $\boldsymbol{J}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U})$ represents an arbitrary loss term for the center word $c = w_t$ and outside word $w_{t+j}$. $\boldsymbol{J}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U})$ could be $\boldsymbol{J}_{\text{naive-softmax}}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U})$ or $\boldsymbol{J}_{\text{neg-sample}}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U})$, depending on your implementation.

Write down three partial derivatives:

(i) $\frac{\partial \boldsymbol{J}_{\text{skip-gram}}(\boldsymbol{v}_c, w_{t-m}, \ldots w_{t+m}, \boldsymbol{U})}{\partial \boldsymbol{U}}$

(ii) $\frac{\partial \boldsymbol{J}_{\text{skip-gram}}(\boldsymbol{v}_c, w_{t-m}, \ldots w_{t+m}, \boldsymbol{U})}{\partial \boldsymbol{v}_c}$

(iii) $\frac{\partial \boldsymbol{J}_{\text{skip-gram}}(\boldsymbol{v}_c, w_{t-m}, \ldots w_{t+m}, \boldsymbol{U})}{\partial \boldsymbol{v}_w}$ when $w \neq c$

Write your answers in terms of $\frac{\partial \boldsymbol{J}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U})}{\partial \boldsymbol{U}}$ and $\frac{\partial \boldsymbol{J}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U})}{\partial \boldsymbol{v}_c}$. This is very simple – each solution should be one line.

***Once you're done:*** *Given that you computed the derivatives of $\boldsymbol{J}(\boldsymbol{v}_c, w_{t+j}, \boldsymbol{U})$ with respect to all the model parameters $\boldsymbol{U}$ and $\boldsymbol{V}$ in parts (a) to (c), you have now computed the derivatives of the full loss function $\boldsymbol{J}_{\text{skip-gram}}$ with respect to all parameters. You're ready to implement* `word2vec`*!*

(i) $\frac{\partial J_{SG}}{\partial U} = \sum_{-m \leq j \leq m} \frac{\partial J(V_c, W_{t+j}, U)}{\partial U}$

(ii) $\frac{\partial J_{SG}}{\partial V_c} = \sum_{-m \leq j \leq m} \frac{\partial J(V_c, W_{t+j}, U)}{\partial V_c}$

(iii) $\frac{\partial J_{SG}}{\partial V_w} = 0$ when $w \neq c$

(c) (2 points) Show time! Now we are going to load some real data and train word vectors with everything you just implemented! We are going to use the Stanford Sentiment Treebank (SST) dataset to train word vectors, and later apply them to a simple sentiment analysis task. You will need to fetch the datasets first. To do this, run `sh get_datasets.sh`. There is no additional code to write for this part; just run `python run.py`.

*Note: The training process may take a long time depending on the efficiency of your implementation and the compute power of your machine* ***(an efficient implementation takes one to two hours)****. Plan accordingly!*

After 40,000 iterations, the script will finish and a visualization for your word vectors will appear. It will also be saved as `word_vectors.png` in your project directory. **Include the plot in your homework write up.** In at most three sentences, briefly explain what you see in the plot. This may include, but is not limited to, observations on clusters and words that you expect to cluster but do not.

We can easily see that familiar words are clustered together female, woman..

we can also see that there are similar vectors between similar compntext

king queen and male female.