Assignment 2 Written Solutions

(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 $\boldsymbol{y}$ (true distribution), $\hat{\boldsymbol{y}}$ (predicted distribution) are vectors and $\hat{\boldsymbol{y}}_o$ is a scalar):

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

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

Cross-Entropy $(y, \hat{y}) = -\sum_{i=1}^{Vocab} y_i \log(\hat{y}_i) = -(y_1 \log \hat{y}_1 + \cdots + y_o \log \hat{y}_o + \cdots + y_{vocab} \log \hat{y}_{vocab})$
But $y_1 = y_2 = \cdots = y_{vocab} = 0 \neq y_o = 1$. So,
Cross-Entropy $(y, \hat{y}) = -\log \hat{y}_o = -\log\big(P(O = o|C = c)\big) = J_{Naive-Softmax}(v_C, o, U)$

(b) (6 points)

(i) Compute the partial derivative of $\boldsymbol{J}_{\text{naive-softmax}}(\boldsymbol{v}_c, o, \boldsymbol{U})$ 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 $\boldsymbol{y}_1$, $\boldsymbol{y}_2$, ...).

(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 the 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.

(i) $J_{Naive-Softmax}(v_C, o, U) = -\log\big(P(O = o|C = c)\big) = -\log\left(\frac{e^{u_o^T v_c}}{\sum_{w \in vocab} e^{u_w^T v_c}}\right)$

$= -\log\big(e^{u_o^T v_c}\big) + \log\big(\sum_{w \in vocab} e^{u_w^T v_c}\big) = -u_o^T v_c + \log\big(\sum_{w \in vocab} e^{u_w^T v_c}\big)$

$\frac{\partial J_{Naive-Softmax}(v_C, o, U)}{\partial v_c} = -u_o^T + \frac{\sum_{w \in vocab} e^{u_w^T v_c} u_w^T}{\sum_{w \in vocab} e^{u_w^T v_c}}$

But $\hat{y}_w = \log\big(P(O = o|C = c)\big) = \frac{e^{u_w^T v_c}}{\sum_{w \in vocab} e^{u_w^T v_c}}$. So,

$\frac{\partial J_{Naive-Softmax}(v_C, o, U)}{\partial v_c} = -u_o^T + \sum_{w \in vocab} \hat{y}_w u_w^T$

We can rewrite u_o^T as $u_o^T = y_1u_1 + y_2u_2 + \cdots + y_ou_o + \cdots y_{vocab}u_{vocab} = \sum_{w \in vocab} y_w\, u_w^T$

as $y = \begin{bmatrix} y_1 = 0 \\ y_2 = 0 \\ y_o = 1 \\ \vdots \\ y_{vocab} = 0 \end{bmatrix}$. In conclusion,

$\frac{\partial J_{Naive-Softmax}(v_C,o,U)}{\partial v_c} = -\sum_{w \in vocab} y_w\, u_w^T + \sum_{w \in vocab} \hat{y}_w\, u_w^T = \sum_{w \in vocab}(-y_w + \hat{y}_w)u_w^T$

Vectorizing this result yields:

$$\frac{\partial J_{Naive-Softmax}(v_C, o, U)}{\partial v_c} = U^T(\hat{y} - y)$$

(ii) When $\hat{y} = y$, which means that the predicted distribution is equal to the true distribution.

(iii) $v_c^{New} = v_c^{Old} - \alpha \frac{\partial J_{Naive-Softmax}(v_C,o,U)}{\partial v_c} = v_c^{Old} - \alpha U(y - \hat{y}) = v_c^{Old} - \alpha Uy + \alpha U\hat{y}$

v_c^{Old} heads towards the direction of the true distribution and away from the direction of the predicted distribution.

(c) (1 point) 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. $\mathbf{u}$). Let's consider a hypothetical downstream task of binary classification of phrases as being positive or negative, where you decide the sign based on the sum of individual embeddings of the words. 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 α. Give examples of words x and y which satisfy the given relation and why would they be affected/not affected due to the normalization?

Mostly, normalization is useful and doesn't take away any useful information. However, when we consider a case like the given one, e.g., $u_x = \alpha u_y$, there will be a problem in normalizing the two vectors. They will have the same orientation but different scales.

An example of that is the words "chair" and "chairs". While they may have similar direction and different scales, the task could be to distinguish the plural nouns from the singular ones. Normalization will only keep orientation information while removing the scale information which will cause the 2 words to be the same and the task will fail.

(d) (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.

$J_{Naive-Softmax}(v_C, o, U) = -u_o^T v_c + \log\left(\sum_{w \in vocab} e^{u_w^T v_c}\right)$

$\frac{\partial J_{Naive-Softmax}(v_C,o,U)}{\partial u_w} = -\frac{\partial}{\partial u_w}(u_o^T v_c) + \frac{e^{u_w^T v_c} v_c}{\sum_{w \in vocab} e^{u_w^T v_c}} = -\frac{\partial}{\partial u_w}(u_o^T v_c) + \hat{y}_w v_c$

When $u_o = u_w$: $\frac{\partial}{\partial u_w}(u_o^T v_c) = \frac{\partial}{\partial u_w}(u_w^T v_c) = v_c$

When $u_o \neq u_w$: $\frac{\partial}{\partial u_w}(u_o^T v_c) = 0$

We can say in general that: $\frac{\partial}{\partial u_w}(u_o^T v_c) = y_w v_C$ where $y_w = \begin{cases} 0, & w \neq o \\ 1, & w = o \end{cases}$

$\frac{\partial J_{Naive-Softmax}(v_C,o,U)}{\partial u_w} = -y_w v_C + \hat{y}_w v_c = (\hat{y}_w - y_w) v_c$

(e) (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 $\frac{\partial \boldsymbol{J}(\boldsymbol{v}_c,o,\boldsymbol{U})}{\partial \boldsymbol{u}_1}$, $\frac{\partial \boldsymbol{J}(\boldsymbol{v}_c,o,\boldsymbol{U})}{\partial \boldsymbol{u}_2}$, $\cdots$, $\frac{\partial \boldsymbol{J}(\boldsymbol{v}_c,o,\boldsymbol{U})}{\partial \boldsymbol{u}_{|\text{Vocab}|}}$. No derivations are necessary, just an answer in the form of a matrix.

$\frac{\partial J_{Naive-Softmax}(v_C,o,U)}{\partial U} = \left[\frac{\partial J_{Naive-Softmax}(v_C,o,U)}{\partial u_1}, \ldots, \frac{\partial J_{Naive-Softmax}(v_C,o,U)}{\partial u_{vocab}}\right]$

But from (d): $\frac{\partial J_{Naive-Softmax}(v_C,o,U)}{\partial u_w} = (\hat{y}_w - y_w) v_c$

$\frac{\partial J_{Naive-Softmax}(v_C,o,U)}{\partial U} = [(\hat{y}_1 - y_1) v_c, \ldots, (\hat{y}_{vocab} - y_{vocab}) v_c] = (\hat{y} - y)^T v_C$

(f) (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]

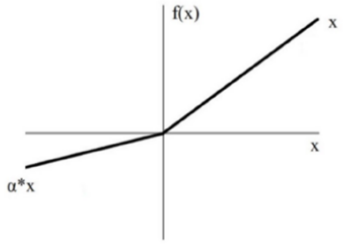

Figure 2: Leaky ReLU

$$f(x) = \begin{cases} \alpha x, & x < 0 \\ x, & x \geq 0 \end{cases}$$

$$\frac{\partial f(x)}{\partial x} = \begin{cases} \alpha, & x < 0 \\ 1, & x \geq 0 \end{cases}$$

(g) (3 points) The sigmoid function is given by Equation 5:

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

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.

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

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

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

$$= \sigma(x)\big(1 - \sigma(x)\big)$$

(h) (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 (d), 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 (g) to help compute the necessary gradients here.

(ii) In the lecture, we learned that an efficient implementation of backpropagation leverages the reuse of previously computed partial derivatives. Which quantity could you reuse (we store the common terms in a single matrix/vector) 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.

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

(i) $J_{neg-sample}(v_C, o, U) = -\log\big(\sigma(u_o^T v_c)\big) - \sum_{s=1}^{K} \log\big(\sigma(-u_{ws}^T v_c)\big)$

$$\frac{\partial J_{neg-sample}(v_C, o, U)}{\partial v_c} = -\frac{\sigma(u_o^T v_c)\big(1-\sigma(u_o^T v_c)\big)}{\sigma(u_o^T v_c)} u_o - \sum_{s=1}^{K} \frac{\sigma(-u_{ws}^T v_c)\big(1-\sigma(-u_{ws}^T v_c)\big)}{\sigma(-u_{ws}^T v_c)} (-u_{ws})$$

$$= -\big(1-\sigma(u_o^T v_c)\big)u_o + \sum_{s=1}^{K} \big(1-\sigma(-u_{ws}^T v_c)\big)u_{ws}$$

$$\frac{\partial J_{neg-sample}(v_C, o, U)}{\partial u_o} = -\frac{\sigma(u_o^T v_c)\big(1-\sigma(u_o^T v_c)\big)}{\sigma(u_o^T v_c)} v_c - 0 = \big(1-\sigma(u_o^T v_c)\big)v_c$$

$$\frac{\partial J_{neg-sample}(v_C, o, U)}{\partial u_{ws}} = 0 - \frac{\sigma(-u_{ws}^T v_c)\big(1-\sigma(-u_{ws}^T v_c)\big)}{\sigma(-u_{ws}^T v_c)} (-v_c) = \big(1-\sigma(-u_{ws}^T v_c)\big)v_c$$

(ii) We can store $\begin{bmatrix} 1-\sigma(u_o^T v_c) \\ 1-\sigma(u_1^T v_c) \\ 1-\sigma(u_2^T v_c) \\ \vdots \\ 1-\sigma(u_{wK}^T v_c) \end{bmatrix}$ as these quantities are calculated in each derivative.

(iii) It computes the gradients using only K samples instead of using the entire vocabulary. It saves both computation time and memory.

(i) (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.

$$J_{neg-sample}(v_C, o, U) = -\log\big(\sigma(u_o^T v_c)\big) - \sum_{s=1}^{K} \log\big(\sigma(-u_{ws}^T v_c)\big)$$

$$= -\log\big(\sigma(u_o^T v_c)\big) - \sum_{\substack{i=1 \\ w_i = w_s}}^{K_1} \log\Big(\sigma(-u_{wi}^T v_c)\Big) - \sum_{\substack{j=1 \\ w_j \neq w_s}}^{K-K_1} \log\Big(\sigma\big(-u_{wj}^T v_c\big)\Big)$$

$$\frac{\partial J_{neg-sample}(v_C, o, U)}{\partial u_{ws}} = 0 - \sum_{\substack{i=1 \\ w_i = w_s}}^{K_1} \frac{\sigma(-u_{ws}^T v_c)\big(1-\sigma(-u_{ws}^T v_c)\big)}{\sigma(-u_{ws}^T v_c)}(-v_c) - 0$$

$$= \sum_{\substack{i=1 \\ w_i = w_s}}^{K_1} \big(1-\sigma(-u_{ws}^T v_c)\big)v_c = K_1\big(1-\sigma(-u_{ws}^T v_c)\big)v_c$$

(j) (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.

(i)

$$\frac{\partial J_{skip-gram}(v_c, w_{t-m}, \ldots, w_{t+m}, U)}{\partial U} = \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \frac{\partial J(v_c, w_{t+j}, U)}{\partial U}$$

(ii)

$$\frac{\partial J_{skip-gram}(v_c, w_{t-m}, \ldots, w_{t+m}, U)}{\partial v_c} = \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \frac{\partial J(v_c, w_{t+j}, U)}{\partial v_c}$$

(iii)

$$\frac{\partial J_{skip-gram}(v_c, w_{t-m}, \ldots, w_{t+m}, U)}{\partial v_w} = 0$$